LINUX

Příručka správce sítě

Kapitola

1

Úvod do vytváření sítí

1.1 Historie

Myšlenka vytváření sítí je pravděpodobně stejně stará jako vlastní telekomunikace. Uvažte lidi žijící v době kamenné, kteří si možná mezi sebou předávali zprávy pomocí bubnů. Dejme tomu, že jeskynní člověk A chtěl pozvat jeskynního člověka B na zábavnou hru, která spočívala v tom, že po sobě házeli kamením, ale žil příliš daleko na to, aby B slyšel jeho obrovský buben. Jaké měl tedy A volby? Mohl 1) dojít za B, 2) sehnat si větší buben nebo 3) požádat C, který žil v polovině vzdálenosti jejich obydlí, aby zprávu předal. V případě poslední možnosti se jedná o vytvoření sítě.

Samozřejmě, že od primitivních snah a prostředků našich předků jsme urazili již dlouhou cestu. Když si dnes chceme domluvit schůzku na sobotní fotbalový zápas, máme počítače, které spolu komunikují prostřednictvím soustavy drátů, optických vláken, mikrovln apod.[1] V následujícím textu se budeme zabývat způsoby a metodami, kterými je to realizováno, opustíme však záležitosti kolem drátů i kolem fotbalu.

V tomto průvodci si popíšeme dva typy sítí: sítě založené na protokolu UUCP a sítě založené na protokolu TCP/IP. Jde o sady protokolů a softwarové balíky, které obstarávají přenos dat mezi počítači. V této kapitole se podíváme na oba typy sítí a probereme si jejich základní principy.

Síť definujeme jako soubor *hostitelů*, kteří spolu mohou vzájemně komunikovat a často se přitom spoléhají na služby vyhrazených hostitelů, přenášejících data mezi jednotlivými účastníky. Hostitelé jsou často počítače, ale neplatí to vždy; někdy tak nazýváme i X-terminály nebo inteligentní tiskárny. Menším seskupením hostitelů říkáme *místa (site)*.

---

1 Původní duch, který se v Evropě stále ještě zjevuje.

Komunikace by nebyla možná bez určitého druhu jazyka nebo kódu. V počítačových sítích se těmto jazykům souhrnně říká *protokoly*. Neměli byste si však představovat písemné protokoly, ale spíše vysoce formalizovaný kód chování, které lze pozorovat například při setkání hlav států. Podobně protokoly používané v počítačových sítích nejsou ničím jiným, než pravidly pro výměnu zpráv mezi dvěma hostiteli.

1.2 Sítě UUCP

UUCP je zkratka spojení Unix-to-Unix Copy. Původně to byl balík programů pro přenos souborů po sériových linkách, plánování těchto přenosů a spouštění programů na vzdálených hostitelích. Od své první implementace koncem sedmdesátých let prodělal tento protokol velké změny, ale co se týče služeb, které nabízí, je stále poněkud nekomfortní. Hlavní využití má stále v sítích WAN založených na vytáčených (dial-up) telefonních linkách.

Protokol UUCP byl vytvořen v Bellových laboratořích v roce 1977, kde měl sloužit ke komunikaci mezi místy podílejícími se na vývoji Unixu. V polovině roku 1978 spojovala tato síť již přes 80 míst. Používala se zde elektronická pošta i vzdálený tisk. Ovšem hlavní využití systému spočívalo v šíření nového softwaru a opravách chyb.[2] Dnes již není protokol UUCP omezen jen na prostředí Unixu. Existují jak volně šiřitelné, tak i komerční porty na různé platformy, včetně AmigaOS, DOS, Atari TOS atd.

Jednou z hlavních nevýhod sítí UUCP je malá šířka pásma. Na jedné straně telefonní vybavení úzce omezuje maximální přenosovou rychlost. Na druhé straně jsou spojení prostřednictvím protokolu UUCP jen zřídka trvalá; hostitelé se místo toho spojují v pravidelných intervalech. Proto většinu času, který zabere doručení poštovní zprávy v síti UUCP, zpráva jen nečinně „dřepí“ na disku nějakého hostitele a čeká na nadcházející navázání spojení.

Navzdory těmto omezením stále funguje po celém světě spousta sítí UUCP, o něž pečují zejména nadšenci, kteří za rozumné ceny nabízí soukromým uživatelům přístup k síti. Hlavním důvodem oblíbenosti protokolu UUCP je skutečnost, že ve srovnání s připojením počítače pevnou linkou je téměř „za babku“. K tomu, abyste z vašeho počítače udělali uzel UUCP, potřebujete jen modem, fungující implementaci protokolu UUCP a jiný uzel UUCP, který bude ochotný vás zásobovat poštou a newsy.

2 Ne že by se od té doby časy příliš změnily…

1.2.1 Jak používat protokol UUCP

Myšlenka stojící v pozadí protokolu UUCP je velice jednoduchá: jak již jeho název napovídá, kopíruje vlastně soubory z jednoho hostitele na druhý. Kromě toho ale umožňuje provádět na vzdáleném hostiteli také určité akce.

Dejme tomu, že váš počítač má povolen přístup k hypotetickému hostiteli jménem **swim** a může na něm spouštět tiskový příkaz `lpr`. Potom byste mohli na hostiteli **swim** vytisknout tuto knihu za pomoci následujícího příkazu zadaného v příkazové řádce:[3]

```
$ uux -r swim!lpr !netguide.dvi
```

Příkaz `uux` ze sady protokolu UUCP naplánuje úlohu pro hostitele **swim**. Tuto úlohu formuluje vstupní soubor `netguide.dvi` a požadavek o předání tohoto souboru příkazu `lpr`. Příznak `-r` říká příkazu `uux`, aby nevolal vzdálený systém ihned, ale aby úlohu pozdržel, dokud nebude při příští příležitosti navázáno spojení. Tomuto postupu říkáme *spooling*.

Další vlastností protokolu UUCP je, že umožňuje doručovat úlohy a soubory přes několik hostitelů, za předpokladu, že tyto spolupracují. Dejme tomu, že hostitel **swim** z předchozích příkladů má UUCP-spojení s hostitelem **groucho**, který spravuje rozsáhlý archiv unixových aplikací. Ke stažení souboru *tripwire-1.0.tar.gz* můžete použít následující příkaz:

```
$ uucp  -mr  swim!groucho!~/security/tripwire-1.0.tar.gz trip.tgz
```

Vytvořená úloha požádá hostitele **swim**, aby získal příslušný soubor od hostitele **groucho** a poslal ho zpět vašemu počítači, přičemž ho protokol UUCP uloží do souboru `trip.tgz` a oznámí vám poštou jeho doručení. Výše uvedený proces proběhne ve třech krocích. Nejprve pošle váš počítač úlohu hostiteli **swim**. Jakmile tento hostitel naváže příští spojení s hostitelem **groucho**, nahraje příslušný soubor. Závěrečným krokem je vlastní přenos souboru z hostitele **swim** na váš počítač.

Nejdůležitější služby, které dnes poskytují sítě založené na protokolu UUCP, jsou elektronická pošta a síťové news. K těmto tématům se ještě vrátíme později, takže nyní si je zde jen nastíníme.

Elektronická pošta – zkráceně e-mail – umožňuje uživatelům předávání zpráv s jinými uživateli pracujícími na vzdálených hostitelích, aniž by bylo nutné znát cestu k těmto hostitelům. Proces směrování zprávy od vašeho počítače k cílovému počítači je plně v režii systému pro správu pošty. V prostředí UUCP jsou poštovní zprávy zpravidla předávány sousednímu hosti-

3 Při použití příkazového interpretu *bash*, GNU Bourne Again Shell, možná bude nutné zrušit význam znaku vykřičník, protože tento znak se zde používá jako znak historie.

teli za pomoci příkazu `rmail`, kterému je předána adresa příjemce a vlastní zpráva. Příkaz `rmail` potom přesměruje zprávu na jiného hostitele atd., dokud nedorazí k cílovému hostiteli. Na doručování zpráv se podrobněji podíváme v kapitole 13.

Síťové news lze nejlépe přirovnat k jistému druhu distribuované BBS (Bulletin Board System – Elektronický konferenční systém). Nejčastěji se tento termín používá pro „usenetové news," což je nejznámější síť pro výměnu news, u které se odhaduje, že má 120 000 účastníků*. Počátky Usenetu se datují do roku 1979, kdy krátce po vydání protokolu UUCP, jakožto součásti Unixu V7, přišli tři absolventi univerzity s myšlenkou všeobecné výměny informací v rámci unixové komunity. Dali dohromady několik skriptů, čímž vznikl první systém síťových news. V roce 1980 se k této síti připojily servery **duke**, **unc** a **phs** a dvě univerzity v Severní Karolíně. I bez toho se však Usenet konečně začal rozrůstat. I když se původně jednalo o síť založenou na protokolu UUCP, dnes se neomezuje pouze na tento jediný typ sítě.

Základní jednotkou informace je zde článek, který lze zaslat do hierarchie diskusních skupin vyhrazených určitému tématu. Většina míst dostává pouze výběr některých diskusních skupin, což se v průměru rovná asi 60 MB článků denně.

Ve světe UUCP jsou newsy posílány prostřednictvím UUCP- spojení tím způsobem, že se vezmou všechny články požadované diskusní skupiny a sbalí se do několika *dávek (batches)*. Tyto se pak pošlou příslušnému počítači, který je rozbalí a dále zpracuje za pomoci příkazu `rnews`.

UUCP je konečně také médiem archivů umožňujících přístup přes vytáčenou linku a nabízejících veřejný přístup. Přístup k nim znamená připojení prostřednictvím protokolu UUCP, přihlášení se jako uživatel-host a stažení souborů z veřejně dostupné archivní oblasti. Tyto účty mají zpravidla jméno a heslo typu **uucp/nuucp** nebo jim podobné.

1.3 Sítě TCP/IP

I když se protokol UUCP hodí pro levné vytáčené síťové linky, v mnoha situacích je metoda store-and-forward (ulož a předej) příliš málo pružná, například v sítích LAN (Local Area Networks). Tyto sítě jsou obvykle tvořeny malým počtem počítačů, umístěných v jedné budově či dokonce na stejném podlaží a vzájemně propojených, čímž vznikne homogenní pracovní prostředí. Typicky budete chtít sdílet soubory v rámci těchto hostitelů nebo na různých počítačích spouštět distribuované aplikace.

* Poznámka korektora: Dnes je to již podstatně více.

Tyto úlohy vyžadují kompletně odlišný přístup k vytváření sítí. Místo přenášení celých souborů společně s popisem úlohy jsou všechna data rozdělena na malé kousky (pakety), které jsou ihned předány cílovému hostiteli, který je zase zpětně spojí. Tomuto typu sítí říkáme *packed-switched* (přenos paketů). Kromě jiného tak lze prostřednictvím sítě spouštět interaktivní aplikace. Cenou je samozřejmě výrazně složitější software.

Řešení, které přijal systém Unix (a také spousta neunixových platforem) se nazývá TCP/IP. V této stati se podíváme na základní pojetí tohoto protokolu.

1.3.1 Úvod do sítí TCP/IP

Protokol TCP/IP byl vyvinut v rámci výzkumného projektu financovaného americkou společností DARPA (Defense Advanced Research Projects Agency) v roce 1969. Jednalo se o experimentální síť zvanou ARPANET, která byla uvedena do provozu v roce 1975, poté co se ukázalo, že by mohla mít úspěch.

V roce 1983 byla standardizována sada protokolů TCP/IP, kterou museli používat všichni hostitelé v této síti. Když se ze sítě ARPANET nakonec vyvinul Internet (přičemž síť ARPANET sama o sobě zanikla v roce 1990), rozšířilo se používání protokolu TCP/IP i v sítích mimo vlastní Internet. Nejpozoruhodnější jsou lokální sítě na bázi Unixu, ale v předvečer nástupu rychlých telefonních zařízení typu IDSN má slibnou budoucnost i jako přenosový protokol pro vytáčené sítě.

Abychom si ukázali konkrétnější využití protokolu TCP/IP, který budeme probírat v následujících statích, vezmeme si za příklad univerzitu GMU (Groucho Marx University), která se nachází někde v Fredlandu. Většina kateder zde má vlastní lokální sítě, některé je sdílejí a jiné jich zase mají více. Všechny jsou vzájemně propojeny a k Internetu jsou připojeny jednou vysokorychlostní linkou.

Dejme tomu, že váš stroj pracující pod Linuxem je připojen k lokální síti unixových hostitelů na katedře matematiky a jmenuje se **erdos**. Budete-li se chtít připojit k hostiteli na katedře fyzyky, který se jmenuje řekněme **quark**, zadáte následující příkaz:

```
$ rlogin quark.physics
Welcome to the Physics Department at GMU
(ttyq2) login:
```

Na výzvu zadáte vaše uživatelské jméno, třeba **andres,** a vaše heslo. Pak se vám spustí příkazový interpret na hostiteli **quark**, se kterým můžete pracovat stejným způsobem, jako byste seděli u konzoly tohoto systému. Právě jste použili jednu z okamžitých interaktivních aplikací, které poskytuje protokol TCP/IP: vzdálené přihlášení.

Jakmile jste připojeni ke stroji **quark**, budete asi chtít spouštět také nějaké aplikace systému X11, například tiskový program nebo prohlížeč postscriptových dokumentů. Aby příslušná aplikace věděla, že chcete mít její okna zobrazena na obrazovce vašeho hostitele, je třeba nastavit proměnnou prostředí *DISPLAY*:

```
$ export DISPLAY=erdos.maths:0.0
```

Když nyní tuto aplikaci spustíte, spojí se místo stroje **quark** s X-serverem a zobrazí všechna svá okna na vaší obrazovce. K tomu je samozřejmě nutné, aby na počítači **erdos** běžel systém X11. Protokol TCP/IP umožňuje hostitelům **quark** a **erdos** vzájemnou výměnu paketů, což budí dojem, že pracujete v jednom systému. Síť je zde takřka transparentní.

Další velice důležitou vlastností sítí TCP/IP je souborový systémy NFS (Network File Systém). Také on přispívá k transparentnosti sítě, protože umožňuje připojovat adresářové struktury z jiných hostitelů, které se pak jeví jako lokální souborové systémy. Například všechny domovské adresáře uživatelů mohou být na centrálním serveru, ze kterého si je připojují všichni ostatní hostitelé v lokální síti. V důsledku toho se uživatelé mohou přihlásit na libovolný stroj a získají vždy stejný domovský adresář. Podobně je možné nainstalovat aplikace, které vyžadují velký diskový prostor (například TEX), pouze na jednom počítači a tyto soubory pak na ostatní počítače exportovat. K souborovému systému NFS se ještě vrátíme v kapitole 11.

Samozřejmě, že to jsou jen příklady využití sítí založených na protokolu TCP/IP. Možnosti jsou takřka neomezené.

Nyní se podrobněji podíváme na způsob, jakým funguje protokol TCP/IP. Jeho znalost vám pomůže lépe pochopit jak a proč máte nakonfigurovat váš počítač. Začneme hardwarem a pomalu se dostaneme až k tomuto protokolu.

1.3.2 Ethernety

Typu hardwaru, který se nejčastěji používá v lokálních sítích, říkáme *Ethernet*. Je tvořen jedním kabelem, ke kterému jsou jednotliví hostitelé připojeni prostřednictvím konektorů, odboček nebo transceiverů. Jednoduché Ethernety nejsou příliš drahé, což je společně s jejich přenosovou rychlostí 10 Mbitů za sekundů důvodem jejich oblíbenosti.

Ethernety existují ve třech provedeních, které nazýváme *tlustý (thick)*, *tenký (thin)* a *kroucená dvojlinka (twisted pair)*. Tlustý a tenký Ethernet používají koaxiální kabel, který se liší šířkou a způsobem, jakým k němu lze hostitele připojit. Tenký Ethernet používá konektor „BNC“ ve tvaru písmene T, který vložíte do kabelu a zapojíte do zadní stěny vašeho počítače. Tlustý Ethernet vyžaduje vyvrtání malé díry do kabelu a připojení transceiveru za pomocí „vampire tap“. Tenký a tlustý Ethernet může být dlouhý maximálně 200 a 500 metrů, a pro-

to se jim také říká 10base-2 a 10base-5. Kroucená dvojlinka je kabel tvořený dvěma měděnými dráty, které se vyskytují také v obyčejných telefonních linkách, ale zpravidla vyžaduje ještě dodatečný hardware. Také se mu říká 10base-T.

I když je připojení hostitele k tlustému Ethernetu poněkud trnité, není přitom třeba odpojovat síť. Pro připojení hostitele k tenkému Ethernetu je třeba alespoň na několik minut přerušit síťové spojení, protože musíte kabel přeříznout, abyste do něho mohli vložit konektor.

Většina lidí upřednostňuje tenký Ethernet, protože je velice levný: síťové karty seženete už za 500 korun a kabel stojí jen několik korun za metr. Nicméně pro rozsáhlejší instalace je vhodnější tlustý Ethernet, aby nemusel být při každém připojování nového hostitele přerušován síťový provoz.

Jednou z nevýhod ethernetové technologie je omezená délka kabelu, což vylučuje jeho použití pro jiné než lokální sítě. Nicméně pomocí repeaterů, mostů a směrovačů může být pospojováno několik ethernetových segmentů. Repeatery prostě jen kopírují signály mezi dvěma nebo více segmenty, takže všechny segmenty dohromady působí dojmem, jako by se jednalo o jediný Ethernet. Dalším omezením Ethernetu je časování, kdy mezi dvěma hostiteli v síti nesmí být více něž čtyři repeatery. Mosty a směrovače jsou propracovanější. Analyzují příchozí data a doručí je pouze v případě, kdy příjemce není připojen lokálním Ethernetem.

Ethernet funguje jako sběrnice, po které může hostitel posílat pakety (nebo *rámce*) až do velikosti 1500 bajtů jinému hostiteli ve stejném Ethernetu. Adresa hostitele je dlouhá šest bajtů a je napevno zakódována do firmwaru jeho ethernetové desky. Takovéto adresy jsou zpravidla zapisovány ve tvaru dvojic hexadecimálních číslic oddělených dvojtečkami, například: **aa:bb:cc:dd:ee:ff**.

Rámec poslaný jednou stanicí vidí všechny připojené stanice, ale pouze cílová stanice si ho vyzvedne a zpracuje. Pokud se ve stejnou dobu pokusí vysílat dvě stanice, doje ke *kolizi*, kterou vyřeší tak, že vysílání přeruší a za nějakou dobu to zkusí znovu.

1.3.3 Další typy hardwaru

U větších instalací, jako je například Groucho Marx University, se zpravidla kromě Ethernetu používá ještě jiný typ vybavení. Na této univerzitě jsou všechny lokální sítě jednotlivých kateder připojeny k univerzitní páteři, což je optický kabel s rozhraním FDDI (*Fiber Distributed Data Interface*). Toto rozhraní používá při přenosu dat úplně jiný přístup, který spočívá v posílání několika *tokenů* a data může vysílat pouze ta stanice, která zachytí nějaký token. Hlavní výhodou rozhraní FDDI je rychlost až 100 Mbps a maximální délka kabelu až 200 km.

Pro dálková síťová spojení se často používají jiné typy vybavení, vycházející ze standardu X.25. Tuto službu nabízí mnoho takzvaných veřejných datových sítí, jakou je v USA například Tymnet nebo v Německu Datex-P.* Standard X.25 vyžaduje speciální hardware, jmenovitě Packet Assembler/Disassembler (*PAD).X.25* definuje sadu vlastních síťových protokolů, ale přesto je často používáno k propojování sítí, které používají protokol TCP/IP nebo jiné. Protože IP pakety nelze jednoduše namapovat na pakety X.25 (a naopak), jsou IP pakety před odesláním jednoduše zapouzdřeny do paketů X.25.

Radioamatéři často používají své vybavení k vzájemnému propojení svých počítačů; tomu říkáme *paketové rádio* nebo tzv. *ham radio*. Protokol, který používají *paketová rádia*, se nazývá AX.25 a byl odvozen od protokolu X.25.

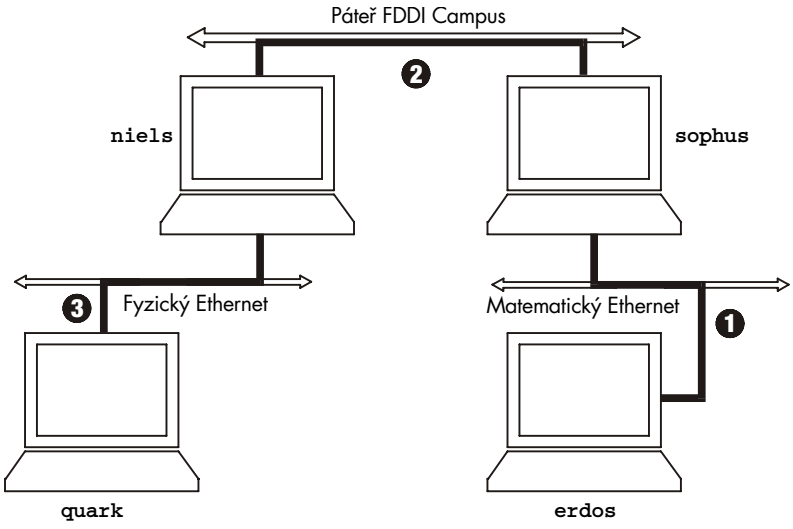
Kromě toho se používají sice pomalé, ale levné sériové linky pro vytáčený (dial-up) přístup. To vyžaduje ještě jeden protokol pro přenos paketů, například SLIP nebo PPP, které si popíšeme dále.

1.3.4 Internetový protokol

Samozřejmě nechcete, aby byla vaše síť omezena jen na Ethernet. V ideálním případě budete chtít používat síť bez ohledu na hardware a počet jednotek, ze kterých je vystavěna. V rozsáhlých sítích, jakou mají například na Groucho Marx University, existuje zpravidla několik oddělených Ethernetů, které je třeba nějakým způsobem pospojovat. Na GMU fungují na katedře matematiky dva Ethernety: první je tvořen rychlými počítači, které využívají profesoři a absolventi, a druhý spojuje pomalejší počítače, na kterých pracují studenti. Obě sítě jsou za pomoci rozhraní FDDI připojeny k univerzitní páteřní síti.

Toto připojení zajišťuje vyhrazený hostitel, takzvaná *brána*, která obsluhuje příchozí a odchozí pakety tím způsobem, že je kopíruje mezi dvěma Ethernety a kabelem s optickými vlákny. Sedíte-li například na katedře matematiky a chcete se z vašeho linuxového stroje připojit k počítači **quark**, který je připojen do lokální sítě na katedře fyziky, nemůže síťový software poslat pakety přímo hostiteli **quark** na katedře fyziky, protože není připojen do stejného Ethernetu. Musí se spolehnout na bránu jakožto na dopravce. Brána (její jméno je **sophus**) pak za pomoci páteřní sítě tyto pakety předá bráně **niels** stejné úrovně na katedře fyziky, která je doručí cílovému počítači. Na obrázku 1.1 je ukázán tok dat mezi bránou **erdos** a **quark** (omlouvám se Guy L. Steeleovi).

* Poznámka korektora: V České republice je to např. společnost Telecom.



Obrázek 1.1
Tři kroky posílání datagramu z počítače **erdos** počítači **quark**

Tomuto schématu doručování dat ke vzdálenému hostiteli se říká *směrování* a paketům v tomto kontextu říkáme *datagramy*. Aby to bylo jednodušší, je výměna datagramů řízena protokolem, který je nezávislý na použitém hardwaru: IP, neboli *Internetový protokol*. Tímto protokolem a otázkami s ním souvisejícími se budeme podrobněji zabývat v kapitole 2.

Hlavní užitek protokolu IP tkví v tom, že spojuje fyzicky rozdílné sítě do jedné zjevně homogenní sítě. Tomu říkáme internetworking (spojování více počítačových sítí do jedné) a výsledné „metasíti“ říkáme *internet*. Všimněte si zde jemného rozdílu mezi slovy *internet* a *Internet*. Druhé z nich označuje konkrétní globální internet.

Samozřejmě, že protokol IP vyžaduje adresové schéma, které je nezávislé na hardwaru. Toho dosáhneme tím, že každému hostiteli přidělíme jedinečné 32bitové číslo nazývané *IP adresa*. IP adresa se zpravidla zapisuje jako čtyři desítková čísla (každé z nich udává 8bitovou část) oddělená tečkami. Například počítač **quark** by mohl mít IP adresu **0x954C0C04**, což bychom zapsali jako **149.76.12.4**. Tomuto formátu zápisu také říkáme *tečková notace*.

Nyní tedy máme tři různé typy adres: nejprve je zde název hostitele, **quark**, potom jeho IP adresa a konečně hardwarová adresa typu 6bajtové ethernetové adresy. Všechny tyto adresy si musí určitým způsobem odpovídat, takže když zadáte `rlogin quark`, může být síťovému softwaru předána IP adresa hostitele **quark** a když protokol IP doručí data do Ethernetu katedry fyziky, musí nějakým způsobem zjistit, která ethernetová adresa této IP adrese odpovídá. Což vypadá poněkud zmateně.

Teď se tím nebudeme zabývat, ale k tomuto tématu se vrátíme v kapitole 2. Nyní nám bude stačit, když si zapamatujeme, že tomuto procesu získávání adresy mapováním názvu hostitele na IP adresu říkáme *rozlišování názvu hostitele* (*hostaname resolution*) a mapování na hardwarovou adresu říkáme *rozlišování adresy* (*address resolution*).

1.3.5 IP přes sériové linky

Na sériových linkách je de facto standardem protokol SLIP, neboli *Serial Line IP*. Modifikovaný typ protokolu SLIP se nazývá CSLIP, neboli *compressed SLIP*, a provádí kompresi IP hlaviček, aby je bylo možné lépe používat v relativně malé šířce pásma, kterou poskytují sériové linky.[4] Odlišným sériovým protokolem je PPP, neboli *Point-to-Point Protocol*. Tento protokol má mnohem více funkcí než protokol SLIP, včetně fáze vyjednávání spojení. Jeho hlavní výhodou oproti protokolu SLIP je, že může přenášet libovolné datagramy, nejenom IP.

1.3.6 Transmission Control Protocol

Samozřejmě, že odesláním datagramů z jednoho hostitele na druhý celý proces nekončí. Přihlásíte-li se k počítači **quark**, chcete, aby bylo spojení mezi vaším procesem *rlogin* na hostiteli **erdos** a příkazovým interpretem na hostiteli **quark** bezpečné. Proto musí odesílatel informaci posílanou z jednoho počítače na druhý rozdělit na pakety a příjemce pak z nich musí znovu sestavit proud znaků. Vypadá to sice jednoduše, ale celý tento proces znamená provedení několika nepříjemných úkolů.

Předně je důležité vědět, že cílem protokolu IP nebylo zajistit bezpečnost. Představte si, že by deset lidí připojených k vaší ethernetové síti začalo stahovat poslední verzi programu Xfree86 z FTP serveru GMU. Síťový provoz, který by tyto operace vyvolaly, by brána nemusela zvládnout, protože je příliš pomalá a příliš těsně svázaná s pamětí. Když teď pošlete z počítače **quark** nějaký paket, nemusí mít brána **sophus** dostatek místa ve vyrovnávací paměti a tím pádem ho nebude moci předat dál. Protokol IP řeší tento problém tak, že příslušný paket zruší. Paket je tak neodvolatelně ztracen. Proto závisí na komunikujících hostitelích, aby kontrolovali integritu a úplnost dat a v případě výskytu chyby přenos opakovali.

K tomu se používá další protokol zvaný TCP, neboli *Transmission Control Protocol*, který vytváří spolehlivou službu nad protokolem IP. Základní vlastností protokolu TCP je, že za pomoci protokolu IP vytváří iluzi jediného spojení mezi příslušnými dvěma procesy na vašem hostiteli a vzdáleném počítači, takže se nemusíte starat jakým způsobem a po jaké trase vaše data ve skutečnosti putují. TCP spojení v podstatě funguje jako dvousměrná roura, do které mohou oba procesy zapisovat i z ní číst. Lze to přirovnat k telefonnímu hovoru.

[4] Protokol SLIP popisuje specifikace RFC 1055. Protokol CSLIP, který provádí kompresi hlaviček, je popisován specifikací RFC 1144.

Protokol TCP rozpozná koncové body takového spojení podle IP adresy příslušných dvou hostitelů a číslo takzvaného *portu* na každém z hostitelů. Na porty je možné se dívat jako na přípojky síťových spojení. Máme-li znovu použít podobnost s telefonním spojením, lze IP adresu přirovnat ke směrovému číslu (směrují čísla na určité město) a čísla portů pak k místním telefonním číslům (směrují čísla na jednotlivé telefony).

V našem příkladu otevře klientská aplikace (`rlogin`) port na hostiteli **erdos** a spojí se s portem 513 na hostiteli **quark**, který odposlouchává server `rlogind`. Tím je navázáno TCP spojení. Za pomoci tohoto spojení provede server `rlogind` autorizaci uživatele a potom spustí příkazový interpret. Standardní vstup a výstup tohoto příkazového interpretu jsou přesměrovány na TCP spojení, takže cokoliv napíšete na vašem počítači, bude proudem TCP přeneseno a předáno na standardním vstupu příkazového interpretu.

1.3.7 User Datagram Protocol

Samozřejmě, že protokol TCP není jediným uživatelským protokolem v sítích TCP/IP. I když je vhodný pro aplikace typu *rlogin,* jeho vysoká režie neumožňuje používat aplikace typu NFS. Místo toho se používá spřízněný protokol UDP, neboli *User Datagram Protocol*. Podobně jako TCP, i protokol UDP dovoluje aplikaci kontaktovat službou na určitém portu na vzdáleném počítači, ale nenaváže s ní spojení. Místo toho lze pomocí tohoto protokolu poslat cílové službě samostatné pakety – odtud je odvozen název tohoto protokolu.

Předpokládejme, že máte připojenu adresářovou strukturu programu **TeX** z centrálního serveru NFS vaší katedry, který se jmenuje ***galois***, a chcete si prohlédnout dokument popisující používání programu LaTeX. Spustíte editor, který nejprve načte celý soubor. Avšak navázání TCP spojení s hostitelem ***galois***, poslání souboru a jeho uvolnění by trvalo příliš dlouho, takže místo toho pošlete hostiteli ***galois*** požadavek, aby poslal příslušný soubor jako dvojici UDP-paketů, což je mnohem rychlejší. Ovšem protokol UDP nebyl vystavěn tak, aby uměl obsloužit ztrátu nebo poškození paketů. Záleží na aplikaci (v tomto případě NFS) aby se o to postarala.

1.3.8 Více o portech

Na porty je možné nahlížet jako na přípojky síťových spojení. Pokud chce nějaká aplikace nabízet určitou službu, připojí sama sebe k určitému portu a čeká na klienty (říká se tomu také *odposlouchávání* portu). Klient, který chce tuto službu využívat, si alokuje nějaký port na lokálním hostiteli a připojí se k příslušnému portu serveru na vzdáleném hostiteli.

Důležitou vlastností portů je, že jakmile bylo navázáno spojení mezi klientem a serverem, může se k portu serveru připojit jiná kopie serveru a odposlouchávat další klienty. To umožňuje například současné přihlášení ke stejnému hostiteli při využití stejného portu

číslo. 513. Protokol TCP je schopen tato spojení rozlišit, protože všechny přicházejí z různých portů nebo hostitelů. Pokud se například dvakrát přihlásíte k hostiteli **quark** z počítače **erdos**, potom bude první klient `rlogin` využívat port 1023 a druhý klient port 1022. Oba však budou připojeni ke stejnému portu 522 na hostiteli **quark**.

Tento příklad ukazuje využití portů jako přístupových bodů, kde se klient připojuje ke specifickému portu, aby získal určitou službu. Aby klient znal správné číslo portu, musí mezi administrátory obou systémů existovat určitá dohoda ohledně přiřazování těchto čísel. U služeb, které jsou široce používány, například `rlogin`, musí být tato čísla spravována centrálně. O to se stará organizace IETF (*Internet Engineering Task Force*), která pravidelně vydává RFC nazvané *Přiřazená čísla* (*Assigned Numbers*). Kromě jiného popisuje čísla portů přiřazená *všeobecně známým službám*. Linux používá službu pro mapování názvů služeb na čísla, která se nazývá `/etc/services`. Je popisována ve stati *Soubory služeb a protokolů*.

Je třeba poznamenat, že i když spojení využívající protokoly TCP i UDP spoléhají na porty, nedochází mezi nimi ke konfliktům. To znamená, že se například TCP port 513 liší od UDP-portu 513. Ve skutečnosti slouží tyto porty jako vstupní body pro dvě různé služby, jmenovitě `rlogin` (TCP) a `rwho` (UDP).

1.3.9 Knihovna socketů

V operačních systémech Unix je software, který obsluhuje veškeré výše popsané úkoly a protokoly, většinou součástí jádra, což platí i pro Linux. Ve světě Unixu je nejběžnějším programovým rozhraním knihovna *Berkeley Socket Library*. Její název je odvozen od oblíbené analogie, která nazírá na porty jako na sockety a připojování k portu chápe jako zapojování. Voláním příkazu (`bind(2)`) je specifikován vzdálený hostitel, přenosový protokol a služba, ke které se program může připojit nebo ji odposlouchávat (pomocí příkazů `connect(2)`, `listen(2)` a `accept(2)`). Knihovna socketů je však obecnější v tom, že poskytuje nejenom třídu socketů založených na protokolu TCP/IP (sockety *AF_INET*), ale také třídu, která obsluhuje spojení s lokálním počítačem (třída *AF_UNIX*). Některé implementace mohou obsluhovat také jiné třídy, jako například protokol XNS (*Xerox Networking Systém*) nebo X.25.

V Linuxu je knihovna socketů součástí standardní knihovny jazyka C zvané *libc*. V současné době podporuje pouze sockety *AF_INET* a *AF_UNIX*, ale pracuje se na podpoře pro síťové protokoly firmy Novell, čehož výsledkem bude začlenění jednoho nebo více tříd socketů tohoto druhu.

1.4 Vytváření sítí v Linuxu

Jakožto výsledek soustředěného úsilí programátorů z celého světa by Linux nemohl existovat bez globální počítačové sítě. Takže nikoho nepřekvapí, že již v raných stádiích vývoje začalo několik lidí pracovat na síťových schopnostech tohoto operačního systému. Implementace protokolu UUCP běžela v Linuxu již od samého začátku a práce na začlenění protokolu TCP/IP začala na podzim roku 1992, když Ross Biro a další vytvořili projekt, který získal označení Net-1.

Poté co Ross v květnu 1993 opustil práci na vývoji tohoto protokolu, začal Fred van Kempen pracovat na nové implementaci, přičemž přepracoval hlavní části kódu. Výsledkem byla verze Net-2. První veřejná verze Net-2d byla hotová v létě 1992 (jako součást jádra verze 0.99.10) a od té doby byla udržována a rozšiřována různými lidmi, zejména Alanem Coxem, který stál u zrodu verze Net-2Debugged. Po důkladném otestování a mnoha vylepšeních kódu přišla po vydání Linuxu verze 1.0 změna názvu na Net-3. Právě tato verze síťového kódu je obsažena v oficiální verzi jádra.

Net-3 nabízí ovladače zařízení pro širokou paletu ethernetových karet, stejně jako SLIP (pro síťový provoz po sériových linkách) a PLIP (pro paralelní linky). Díky Net-3 má Linux implementaci protokolu TCP/IP, která se chová velmi dobře v prostředí lokální sítě, dobu provozu, která překonává některé komerční Unixy pro PC. Vývoj teď směřuje k zajištění nezbytné stability pro běh na internetových hostitelích.

Kromě těchto výhod probíhá několik projektů, které zvýší univerzálnost Linuxu. Ovladač pro protokol PPP (point-to-point protokol je nový způsob síťového přenosu po sériových linkách) je ve stádiu beta-verze a ovladač pro protokol X.25 pro amatérské rádio existuje ve verzi alfa. Alan Cox také implementoval ovladač pro novellovský protokol IPX, ale úsilí o vytvoření kompletní sady nástrojů kompatibilních s produkty firmy Novell bylo nyní pozastaveno, protože firma Novell nepříliš ochotně poskytuje nezbytnou dokumentaci. Dalším velice slibným projektem je *samba*, volně šiřitelný server NetBIOS pro Unices, který napsal Andrew Tridgell.[5]

1.4.1 Různé směry vývoje

Mezitím Fred pokračoval ve vývoji a přešel na verzi Net-2e, která měla přepracovanou strukturu síťové vrstvy. V době psaní této knihy byla Net-2e stále ve stádiu beta-verze. Největším přínosem Net-2e je zařazení rozhraní DDI (*Device Driver Interface*). Toto rozhraní nabízí jednotný přístup a konfiguraci všech síťových zařízení a protokolů.

5 NetBIOS je protokol, na kterém jsou založeny aplikace typu *lanmanager* a Windows for Workgroups.

Další implementace protokolu TCP/IP pochází od Matthiase Urlichse, který napsal ovladač ISDN pro Linux a FreeBSD. Za tím účelem integroval do jádra Linuxu část síťového kódu BSD.

Lze předpokládat, že v brzké době se objeví verze Net-3.* Alan v současné době pracuje na implementaci protokolu AX.25, který používají ham-radioamatéři. Plánovaný „modulový" kód jádra nepochybně vnese nové podněty i do síťového kódu. Moduly umožní připojování ovladačů k jádru za běhu.

I když se tyto různé síťové implementace snaží poskytovat stejné služby, existují mezi nimi na úrovni jádra a ovladačů velké rozdíly. Proto nelze nakonfigurovat systém, na kterém běží jádro Net-2e za pomocí utilit z Net-2d nebo Net-3 a naopak. To se ale týká jen příkazů, které úzce spolupracují s jádrem; aplikace a běžné síťové příkazy, jakými jsou například `rlogin` nebo `telnet`, fungují na obou.

Přesto však by vás všechny tyto různé síťové verze neměly trápit. Pokud se nepodílíte na aktivním vývoji, nemusí vás zajímat, jakou verzi TCP/IP kódu používáte. Oficiální vydání jádra budou vždy obsahovat sadu síťových nástrojů, které jsou kompatibilní se síťovým kódem obsaženým v jádru.

1.4.2 Kde získáte kód

Poslední verzi síťového kódu Linuxu lze získat prostřednictvím anonymního FTP z různých míst. Oficiální FTP server pro Net-3 je **sunacm.swan.ac.uk**, který je zrcadlen na **sunsite.unc.edu**. Poslední verze oprav a spustitelných souborů pro Net-2e najdete na **ftp.aris.com**. Síťový kód odvozený od BSD, jehož autorem je Matthias Urlichs, najdete na **ftp.ira.uka.de** v adresáři `/pub/system/linux/netbsd`.

Poslední verze jádra lze získat na adrese **nic.funet.fi** v adresáři `/pub/OS/Linux/PEOPLE/Linux`, jejíž zrcadla jsou **sunsite** a **tsx-11.mit.edu**.**

1.5 Údržba vašeho systému

V průběhu celé knihy se budeme zabývat především otázkami instalace a konfigurace. Správa však znamená mnohem víc – po zavedení služby je třeba ji také udržovat v provozu. Většina služeb vyžaduje jen nepatrnou údržbu, ale u některých z nich, jako je pošta a news, vyžaduje zachování aktuálnosti systému provádění rutinních úkolů. Tyto úkoly budeme probírat v dalších kapitolách.

* Poznámka korektora: Tato verze dnes již existuje.

** Poznámka korektora: V poslední době je oficiálním archívem zdrojových textů jádra server **ftp.kernel.org.**

Absolutní minimum údržby spočívá v pravidelné kontrole systémových a aplikačních log-souborů, zda neobsahují chybové stavy nebo neobvyklé události. Běžně se to dělá za pomoci dvojice administrativních skriptů, které se pravidelně spouští z `cron`. Tyto skripty obsahují distribuce zdrojových kódů některých hlavních aplikací, jako je například `smail` nebo C news. Je třeba si je jen upravit tak, aby vyhovovaly vašim potřebám a preferencím.

Výstup každé úlohy `cron` by měl být poslán poštou na administrativní účet. Mnoho aplikací implicitně posílá chybové zprávy, statistiky využití nebo souhrny log-souborů na účet **root**. To má smysl, pokud se často přihlašujete jako uživatel **root**, ale výhodnější je za pomocí poštovního aliasu, který popisujeme v kapitole 14, přesměrovat poštu pro uživatele **root** na váš osobní účet.

Bez ohledu na to, jak pečlivě váš systém nastavíte, se v důsledku existence Murphyho zákonů zaručeně vynoří nějaký problém. Proto znamená údržba systému i neustálou připravenost řešit stížnosti. Lidé zpravidla očekávají, že správce systému bude k zastižení přinejmenším prostřednictvím účtu **root**, ale pro kontaktování lidí, kteří jsou odpovědní za určitý aspekt údržby, jsou běžně používány také jiné adresy. Například stížnosti na špatně fungující poštu budou adresovány uživateli **postmaster** a problémy se systémem news je možné oznamovat na adresách **newsmaster** nebo **usenet**. Pošta na účet **hostmaster** by měla být přesměrována člověku, který má na starosti základní síťové služby, a pokud provozujete jmenný server (name-server), také jmennou službu DNS.

1.5.1 Bezpečnost systému

Dalším velice důležitým aspektem správy systému v síťovém prostředí je ochrana systému a uživatelů před vetřelci. Nedbale spravované systémy jsou vhodným cílem pro škodolibé uživatele, jejichž rozsah útoků je poměrně široký: od uhodnutí hesla až po sledování Ethernetu a způsobené škody mohou sahat od padělaných poštovních zpráv až po ztrátu dat nebo narušení soukromí vašich uživatelů. Když budeme hovořit o souvislostech, za kterých k nim může dojít, zmíníme se o některých konkrétních problémech a běžných ochranách proti nim.

Tato stať probírá některé příklady a základní postupy zajištění systémové bezpečnosti. Samozřejmě, že zde uvedená témata nemohou důkladně postihnout všechny bezpečností problémy, s nimiž se můžete setkat; slouží především k ilustraci problémů, které se mohou vyskytnout. Proto je nutné přečíst si nějakou dobrou knihu pojednávající o bezpečnosti, zejména pak v systému propojeném do sítě. Doporučujeme knihu „Practical UNIX Security“, jejímž autorem je Simon Gafinkel (viz [Spaf93]).

Základem bezpečnosti systému je dobrá systémová administrace. Ta zahrnuje kontrolu vlastnictví a přístupových práv ke všem životně důležitým souborům a adresářům, monitorování používání privilegovaných účtů atd. Program COPS například ověří, zda váš systém souborů

(file system) a běžné konfigurační soubory neobsahují neobvyklá přístupová práva nebo jiné anomálie. Také je rozumné používat sadu programů pro správu hesel, které prosadí jistá pravidla pro zadávání hesel takovým způsobem, aby bylo těžké je uhodnout. Stínová sada programů pro správu hesel například vyžaduje, aby bylo heslo dlouhé minimálně pět písmen a obsahovalo, kromě malých a velkých písmen, i číslice.

Při zpřístupňování určité služby přes síť jí nezapomeňte přidělit „nejnižší práva“, což znamená, že jí nedovolíte provádět věci, které pro své fungování nepotřebuje. Programům byste měli například přidělit UID **root** nebo některý jiný privilegovaný účet, jen pokud ho opravdu potřebují. Také v případě, kdy chcete používat některou službu jen pro omezenou aplikaci, ji neváhejte nakonfigurovat s co největším omezením práv, jak to dovolí vaše aplikace. Chcete-li například povolit bezdiskovým hostitelům zavádění z vašeho počítače, musíte zprovoznit službu TFTP (trivial file transver service), aby si tito hostitelé mohli z adresáře `/boot` nahrát základní konfigurační soubory. Pokud byste ale používali službu TFTP bez omezení, umožnili byste komukoliv na světě nahrát si z vašeho systému libovolný soubor, ke kterému má přístupová práva pro čtení. Proč tedy neomezit služby TFTP jen na adresář `/boot`?[6]

Při stejném směru myšlení možná budete chtít omezit určité služby poskytované určitými hostiteli, řekněme z vaší lokální sítě. V kapitole 9 si představíme nástroj `tcpd`, který to umožňuje pro různé síťové aplikace.

Dále je nutné se vyhnout „nebezpečným“ aplikacím. Samozřejmě, že každý software, který používáte, může být svým způsobem nebezpečný, protože může obsahovat chyby, jichž mohou chytří lidé využít k získání přístupu k vašemu systému. K takovýmto věcem dochází, a proto neexistuje úplná ochrana. Tento problém se týká volně šiřitelného softwaru stejně jako komerčních produktů.[7] Nicméně programy, které vyžadují speciální přístupová práva jsou dědičně více nebezpečné než jiné, protože každá díra v nich může mít drastické důsledky.[8] Pokud instalujete program setuid pro síťové účely, pak dávejte dvojnásobný pozor na to, abyste nezapomněli na nic z dokumentace a náhodou tak nevytvořili bezpečnostní trhlinu.

Nikdy nelze vyloučit, že vaše opatření selžou, bez ohledu na to, jak opatrní jste byli. Proto byste měli zajistit včasné odhalení vetřelců. Kontrola systémových log-souborů je dobrým výchozím bodem, ale vetřelec bude zřejmě natolik chytrý, že po sobě smaže všechny zřetelné stopy. Existují však nástroje typu `tripwire`[9], které umožňují kontrolovat, zda nedošlo ke změně obsahu nebo přístupových práv životně důležitých systémových souborů. Program

6 K tomuto tématu se ještě vrátíme v kapitole 9.

7 Existovaly jisté komerční subjekty, kterým jste zaplatili spoustu peněz za setuid-skripty, jež využitím jednoduchého standardního triku dovolovaly uživatelům získat práva **root**.

8 V roce 1988 způsobil červ RTM problémy velké části Internetu tím, že zneužil díry v programu *sendmail*. Tato bezpečnostní díra již byla od té doby odstraněna.

9 Jeho autorem je Gene Kim a Gene Spafford.

`tripwire` provede různé kontrolní součty nad těmito soubory a uloží je do databáze. Později jsou kontrolní součty vypočítány znovu a porovnány s hodnotami uloženými v databázi, čímž se zjistí jakékoliv případné modifikace.

1.6 Přehled následujících kapitol

Několik dalších kapitol se bude zabývat konfigurací Linuxu pro sítě TCP/IP a provozem některých základních aplikací. Dříve než se pustíme do úpravy souborů a věcí okolo si v kapitole 2 podrobněji popíšeme protokol IP. Pokud již ovládáte způsob, jakým funguje IP směrování a jak probíhá rozpoznávání adres, můžete tuto kapitolu přeskočit.

Kapitola 3 se zabývá základními spornými body konfigurace, jako je kompilace jádra a nastavení ethernetové karty. Konfigurace sériových portů je probírána v samostatné kapitole, protože se netýká jen protokolu TCP/IP, ale také protokolu UUCP.

Kapitola 5 vám pomůže zprovoznit váš počítat v síti TCP/IP. Obsahuje rady k instalaci samostatných hostitelů, kteří mají povolenu jen zpětnou vazbu, a hostitelů připojeným k Ethernetu. Také se zde seznámíte s několika užitečnými nástroji, jichž lze využít při testování a ladění vašeho nastavení. Další kapitola probírá konfiguraci rozpoznávání hostitelů a vysvětluje jak nastavit jmenný server.

Potom následují dvě kapitoly, které se zabývají konfigurací a používáním protokolů SLIP a PPP. Kapitola 7 vysvětluje jak navázat spojení pomocí protokolu SLIP a najdete zde také podrobný popis nástroje `dip`, který umožňuje zautomatizovat většinu nezbytných kroků. Kapitola 8 popisuje protokol PPP a démona `pppd`, který k němu potřebujete.

Kapitola 9 je krátkým úvodem do zavádění některých nejdůležitějších síťových aplikací, například `rlogin`, `rcp` atd. Dozvíte se zde také, jak jsou za pomocí *ihetd* spravovány služby a jak lze určité služby, u kterých je důležitá bezpečnost, omezit jen na důvěryhodné hostitele.

V následujících dvou kapitolách je probírán systém NIS, Network Information System, a NFS, Network File System. NIS je užitečný nástroj pro šíření administrativních informací typu uživatelských hesel v lokální počítačové síti. NFS umožňuje sdílet systémy souborů mezi různými hostiteli v síti.

Kapitola 12 je obsáhlým úvodem do správy Taylor UUCP-sady programů protokolu UUCP, která je zdarma.

Zbytek knihy je věnován podrobnému popisu elektronické pošty a usenetových news. V kapitole 13 se dozvíte základní pojmy týkající se elektronické pošty – jak vypadá poštovní adresa a jak systém pro obsluhu pošty zajistí, aby se vaše zpráva dostala k adresátovi.

Kapitoly 14 a 15 popisují nastavení programů `smail` a `sendmail`, dvou agentů pro přenos pošty, jichž lze v Linuxu využívat. Tato kniha popisuje oba dva, protože instalace agenta `smail je` pro začátečníka jednodušší, zatímco agent `sendmail` si lze lépe přizpůsobit.

Kapitoly 16 a 17 objasňují způsob, jakým jsou na Usenetu spravovány síťové news a jak lze nainstalovat C news, oblíbený softwarový balík pro správu usenetových news. Kapitola 18 stručně vysvětluje nastavení démona NNTP, který vaší síti zprostředkuje přístup pro čtení news. A konečně v kapitole 19 se dozvíte, jak nastavit a udržovat různé programy pro čtení news.

Kapitola

2

Problémy sítí na bázi TCP/IP

Nyní se podíváme na problémy, se kterými se můžete setkat při připojení vašeho linuxového počítače k síti, jež využívá protokol TCP/IP. Vyřešíme problémy týkající se IP adres, názvů hostitelů a některé problémy směrování. Základy, které získáte v této kapitole, se vám budou hodit při konfigurování linuxového počítače. Další kapitoly se budou zabývat nástroji, které k tomu budete potřebovat.

2.1 Síťová rozhraní

Aby se vyřešila rozmanitost zařízení, která mohou být používána v síťovém prostředí, definuje protokol TCP/IP abstraktní *rozhraní*, přes které je přistupováno k hardwaru. Toto rozhraní nabízí množinu operací, která je stejná pro všechny typy hardwaru a zabezpečuje posílání a přijímání paketů.

Každé periferní zařízení, které chcete použít v síti, musí mít v jádru operačního systému příslušné rozhraní. Například v Linuxu se rozhraní Ethernet nazývají *eth0* a *eth1* a rozhraní SLIP pak *sl0*, *sl1* atd. Tyto názvy rozhraní se používají při konfiguraci, když chcete v jádru systému pojmenovat konkrétní fyzické zařízení. Jinde nemají žádný význam.

Aby bylo rozhraní použitelné v sítích na bázi TCP/IP, musí mu být přiřazena IP adresa, která slouží jako identifikace při komunikaci se zbytkem světa. Tato adresa je odlišná od názvu rozhraní, o kterém jsme mluvili výše; pokud bychom rozhraní přirovnali ke dveřím, pak adresa odpovídala štítku, který je na nich připevněn.

Samozřejmě lze nastavovat i další parametry; jedním z nich je maximální velikost datagramů, které mohou být zpracovány danou částí hardwaru. Tento parametr se také nazývá *maximální přenosová jednotka* (Maximal Transfer Unit), neboli MTU. Další atributy budou představeny později.

2.2 IP adresy

Již v předchozí kapitole jsme se zmínili o tom, že adresy, kterým rozumí síťový protokol IP, jsou 32bitová čísla. Každému počítači musí být vzhledem k síťovému prostředí přiděleno jedinečné číslo. Pokud provozujete lokální síť, která s ostatními sítěmi nekomunikuje na bázi protokolu TCP/IP, můžete tato čísla přidělit podle vlastního uvážení. Nicméně systémům připojeným k Internetu jsou čísla přidělována centrálně, konkrétně síťovým informačním centrem (Network Information Center), zkráceně NIC.[1]

IP adresy jsou pro přehlednost rozděleny do čtyř 8bitových čísel, kterým říkáme *oktety*. Například adresa **quark.physics.groucho.edu** má IP adresu **0x954C0C04**, která je zapsána jako **149.76.12.4**. Tomuto formátu se také někdy říká *tečková notace*.

Dalším důvodem této symboliky je rozdělení IP adres na číslo *sítě*, které je obsaženo v levých dvou oktetech, a na číslo *hostitele*, které je zapsáno v pravých dvou oktetech. Když požádáte centrum NIC o přidělení IP adres, nebude vám přidělena adresa pro každého hostitele, kterého hodláte používat. Místo toho obdržíte jediné číslo sítě a pak na základě vlastního uvážení můžete přidělovat hostitelům ve vaší síti všechny IP adresy v rámci získaného rozsahu IP adres.

V závislosti na velikosti sítě je nutné mít i různě velkou část hostitele. Protože mají různí uživatelé různé nároky na velikost sítí, existuje několik tříd sítí, které definují různá rozdělení IP adres.

Třída A Třída A obsahuje sítě od **1.0.0.0** do **127.0.0.0**. Číslo sítě je obsaženo v prvním oktetu. Takto je k dispozici 24bitová část hostitele, která umožňuje až 1,6 milionů hostitelů.

Třída B Třída B obsahuje sítě od **128.0.0.0** do **191.255.0.0**; číslo sítě je obsaženo v prvních dvou oktetech. To umožňuje 16 320 sítí, z nichž každá může mít až 65 024 hostitelů.

[1] Často vám jsou IP-adresy přidělovány poskytovatelem, od kterého si IP-spojení koupíte. Můžete však také požádat o přidělení IP-adresy pro vaši síť přímo NIC. Provedete to tak, že pošlete e-mail na adresu **hostmaster@internic.net**.

Třída C Třída C zahrnuje sítě od **192.0.0.0** do **223.255.255.0**, kde číslo sítě je obsaženo v prvních třech oktetech. To umožňuje vytvořit téměř 2 miliony sítí s až 254-mi hostiteli.

Třída D, E a F Adresy, které spadají do intervalu od **224.0.0.0** do **254.0.0.0,** jsou buď experimentální, nebo jsou rezervovány pro budoucí použití. Nespecifikují žádnou síť.

Vrátíme-li se zpět k příkladu z minulé kapitoly, zjistíme, že adresa **149.76.12.4**, což je adresa **quark**, spadá do třídy sítí B **149.76.0.0** a odkazuje na hostitele **12.4**.

Ve výše uvedeném seznamu jste si možná všimli, že v každém oktetu nebyly v příslušné části hostitele povoleny všechny hodnoty. To proto, že hostitelé, jejichž čísla oktetů jsou rovna **0** nebo **255,** jsou rezervováni pro speciální účely. Adresa, ve které jsou všechny bity hostitelské části nulové, odkazuje na síť a adresa, v níž jsou všechny bity hostitelské části rovny jedné, se nazývá vysílací adresa (broadcast address). Vysílací adresa současně odkazuje na všechny hostitele v konkrétní síti. Tedy adresa **149.76.255.255** není koréktní hostitelská adresa, ale odkazuje na všechny hostitele v síti **149.76.0.0**.

Dále existují ještě dvě rezervované síťové adresy, a to **0.0.0.0** a **127.0.0.0**. První z nich se nazývá *implicitní směr* (default route), druhá se nazývá *zpětnovazebná adresa* (loopback address). Implicitní směr souvisí se způsobem směrování IP datagramů, které budeme probírat dále.

Síť **127.0.0.0** je rezervována pro místní IP dopravu k vašim hostitelům. Adresa **127.0.0.1** je obvykle přiřazena speciálnímu rozhraní na vašem hostiteli, které se někdy také nazývá *zpětnovazebné rozhraní* (loopback interface) a chová se jako uzavřený obvod. Každý IP paket předaný protokolem TCP nebo UDP je vrácen zpět, jakoby právě přišel z nějaké sítě. To umožňuje vyvíjet a testovat síťový software, aniž byste používali „skutečnou“ síť. Další užitečnou aplikací je případ, kdy chcete použít síťový software na samostatném hostiteli. Není to zase tak neobvyklé, jak by se mohlo zdát; například mnoho systémů UUCP nemá vůbec žádné IP propojení, ale přesto chtějí provozovat systém news INN. Aby systém INN správně fungoval pod Linuxem, vyžaduje zpětnovazebné rozhraní.

2.3 Rozlišování adres

Když teď víte, jak se vytváří IP adresy, bude vás asi zajímat, jak se s jejich pomocí v Ethernetu adresují různí hostitelé. Protokol Ethernet identifikuje hostitele číslem složeným ze šesti oktetů, které nemá nic společného s IP adresou.

Proto potřebujeme mechanismus, jenž ethernetovým adresám přiřadí IP adresy. Tento mechanismus se nazývá protokol pro rozlišování adres (Address Resolution Protocol), zkráceně ARP, ktrý není omezen pouze na Ethernet, ale používá se i u jiných typů sítí, příkladem je ham-radio. Idea protokolu ARP je stejná jako když hledáte pana X. Představte si tlačenici asi 150-ti lidí, kteří budou chodit pořád dokola, volat jeho jméno a spoléhat na to, že jim v případě, že je přítomen, odpoví.

Když chce ARP najít ethernetovou adresu odpovídající příslušné IP adrese, použije tzv. „vysílání“ (broadcasting), při kterém je datagram současně adresován na všechny stanice v síti. Vyslaný datagram, který odešle protokol ARP, obsahuje dotaz na příslušnou IP adresu. Každý hostitel, který přijímá, ji porovná se svou vlastní IP adresou a pokud souhlasí, vrátí dotazujícímu se hostiteli odpověď pomocí protokolu ARP. Nyní může dotazující se hostitel vytáhnout z odpovědi ethernetovou adresu zasilatele.

Možná se ptáte, jak hostitelé vědí, na kterém z miliard světových Ethernetů lze požadovaného hostitele nalézt a proč to má být zrovna Ethernet? Všechny tyto otázky souvisí se směrováním, tedy nalezením fyzického místa hostitele v síti. To bude předmětem dalšího výkladu.

Podívejme se nyní na protokol ARP. Když hostitel získá ethernetovou adresu, uloží ji do vyrovnávací paměti protokolu ARP, takže když bude dotyčnému hostiteli posílat zpět nějaký datagram, nemusí se na ni znovu dotazovat. Tuto informaci by však nebylo rozumné uchovávat příliš dlouho; ethernetová karta vzdáleného hostitele může být například z důvodu technických problémů vyměněna, čímž by byla data protokolu ARP neplatná. Abychom si vynutili další dotazy na IP adresu, dochází po určité době k mazání vstupů z vyrovnávací paměti protokolu ARP.

Někdy je také nutné nalézt IP adresu, která odpovídá určité ethernetové adrese. To je případ, kdy se bezdiskový počítač pokouší zavádět operační systém ze síťového serveru. V lokálních sítích je to docela běžná situace. Bezdiskový klient však o sobě nemůže poskytnout skoro žádné informace – kromě své ethernetové adresy. Jediné co může udělat je poslat spouštěcímu serveru zprávu se žádostí o sdělení jeho IP adresy. Pro tento účel existuje další protokol, který se nazývá Protokol pro zpětné rozlišení adres (*Reverse Adress Resolution Protocol*), zkráceně RARP. Společně s protokolem BOOTP slouží k definování procedury zavádění operačního systému bezdiskových klientů v síti.

2.4 Směrování IP

2.4.1 Sítě IP

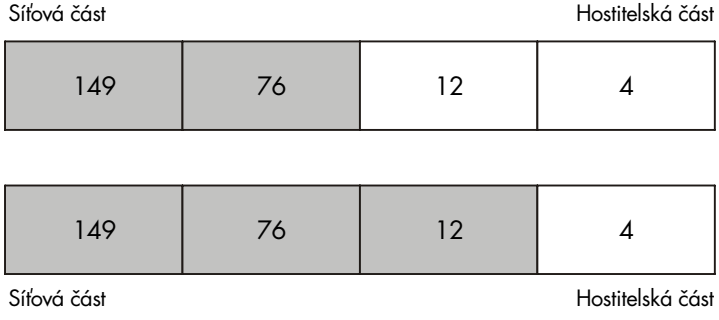
Když někomu píšete dopis, uvedete obvykle na obálce kompletní adresu, která obsahuje název země, stát, kraj, PSČ atd. Poté co ho vhodíte do poštovní schránky, ho pošta doručí na cílové místo: pošle ho do vyznačené země, zde ho národní pošta odešle do příslušného kraje atd. Výhoda tohoto hierarchického schématu je poměrně zřejmá: Ať už pošlete dopis kamkoliv, bude vždy místní poštmistr zhruba vědět směr, kterým má dopis poslat dál, a přitom se nemusí starat o to, po jaké trase poputuje dopis v rámci cílové země.

Sítě IP jsou strukturovány obdobným způsobem. Celý Internet se skládá z mnoha sítí, kterým říkáme *autonomní systémy* (autonomous systems). Každý takovýto systém provádí veškeré směrování v rámci svých členských hostitelů, takže se úkol doručení datagramu zredukuje na hledání cesty k síti cílového hostitele. To znamená, že jakmile datagram zachytí *libovolný* hostitel v cílové síti, bude další proces prováděn výhradně touto sítí.

2.4.2 Podsítě

Tato struktura vyplývá z rozdělení IP adres na část hostitelskou a část síťovou, viz výše uvedený popis. Cílová síť je implicitně odvozena ze síťové části IP adresy. Proto by se hostitelé, kteří mají identická síťová IP čísla, měli nacházet ve stejné síti, což platí i obráceně.[2]

Obdobné schéma má smysl i *uvnitř* sítě, protože vlastní síť se může skládat ze stovek menších sítí, kde jsou nejmenšími jednotkami fyzické sítě, například Ethernety. Protokol IP umožňuje rozdělit sít IP na několik *podsítí*.



Obrázek 2.1
Podsítě sítě třídy B

[2] Nicméně autonomní systémy jsou o něco obecnější. Mohou se skládat z více než jedné sítě IP.

Podsíť na sebe přebírá (od sítě IP, které je součástí) odpovědnost za doručení datagramů pro daný rozsah IP adres. K její identifikaci slouží síťová část IP adresy, jako tomu bylo u tříd A, B nebo C. Nyní je však síťová část rozšířena tak, aby obsahovala i některé bity z hostitelské části. Počet bitů, které jsou interpretovány jako číslo podsítě, udává tzv. *maska podsítě* (*subnet mask*) neboli *síťová maska* (*netmask*). Jedná se také o 32bitové číslo, které určuje bitovou masku části sítě IP adresy.

Příkladem takové sítě je školní síť Groucho Marx University. Má síťové číslo třídy B **149.76.0.0**, a proto je její síťová maska **255.255.0.0**.

Vnitřně je školní síť GMU složena z několika menších sítí, které tvoří například lokální sítě různých kateder. Takže rozsah IP adres je rozložen na 254 podsítí, od adresy **149.76.1.0** po adresu **149.76.254.0**. Například katedra teoretické fyziky má přidělenu adresu **149.76.12.0**. Páteří školní sítě je v pravém slova smyslu síť s adresou **149.76.1.0**. Tyto podsítě sdílí stejné číslo sítě IP, zatímco třetí oktet slouží k jejich rozlišení. Používají tedy masku podsítě **255.255.255.0**.

Obrázek 2.1 ukazuje rozdílnou interpretaci čísla **149.76.12.4**, což je adresa **quark.** Je-li v prvním případě považována adresa za adresu běžné sítě třídy B, pak ve druhém případě je považována za adresu podsítě.

Je dobré vědět, že vytváření podsítí slouží pouze k *vnitřnímu dělení* sítě. Podsítě jsou generovány vlastníkem sítě (nebo jejími správci). Sítě jsou často vytvářeny kvůli hranicím, například fyzickým (mezi dvěma Ethernety), administrativním (mezi dvěma katedrami) nebo geografickým, a pravomocemi nad těmito podsítěmi je pověřena nějaká kontaktní osoba. Tato struktura však odráží pouze vnitřní chování sítě a pro okolní svět je neviditelná.

2.4.3 Brány

Vytváření podsítí nemá jen organizační výhody, je často přirozeným důsledkem hranic hardwaru. „Výhled“ hostitele dané fyzické sítě, jako například Ethernetu, je velmi omezující: jedinými hostiteli, se kterými lze komunikovat přímo, jsou hostitelé v dané síti. Ke všem ostatním hostitelům může být přistupováno jen s pomocí tzv. *bran*. Brána je hostitel, který je současně spojen se dvěma nebo více fyzickými sítěmi a který je nastaven pro přenášení paketů mezi nimi.

Aby protokol IP poznal, zda se hostitel nachází v příslušné lokální fyzické síti, musí různé fyzické sítě náležet k odlišným sítím IP. Například síťové číslo **149.76.4.0** je rezervováno pro hostitele lokální sítě katedry matematiky. Když se posílá datagram na adresu **quark**, síťový software na serveru **erdos** okamžitě z IP adresy **149.76.12.4** pozná, že cílový hostitel je připojen k jiné fyzické síti, a proto je dosažitelný pouze pomocí brány (implicitní bránou je **sophus**).

Vlastní brána **sophus** je propojena se dvěma rozdílnými podsítěmi: s katedrou matematiky a s páteří školní sítě. Ke každé z nich přistupuje za pomoci různých rozhraní *eth0*, resp. *fddi0*. Jakou adresu mu ale přidělíme? Máme mu dát adresu podsítě **149.76.1.0** nebo **149.76.4.0**?

Odpověď zní: obě. Při komunikaci s hostitelem v lokální síti katedry matematiky by měla brána **sophus** použít IP adresu **149.76.4.1** a při komunikaci s hostitelem v páteřní síti by měla použít adresu **149.76.1.4**.

Bráně je tedy přidělena jedna IP adresa pro každou síť s níž je spojena. Tyto adresy – společně s odpovídajícími síťovými maskami – jsou spojeny s rozhraním, ke kterému přistupuje podsíť. Mapování rozhraní a adres v rámci brány **sophus** by tedy mělo vypadat následovně:

Rozhraní	Adresa	Síťová maska
Eth0	**149.76.4.1**	**255.255.255.0**
Fddi0	**149.76.1.4**	**255.255.255.0**
Lo	**127.0.0.1**	**255.0.0.0**

Poslední řádek popisuje zpětnovazebné rozhraní *lo*, které bylo popsáno výše.

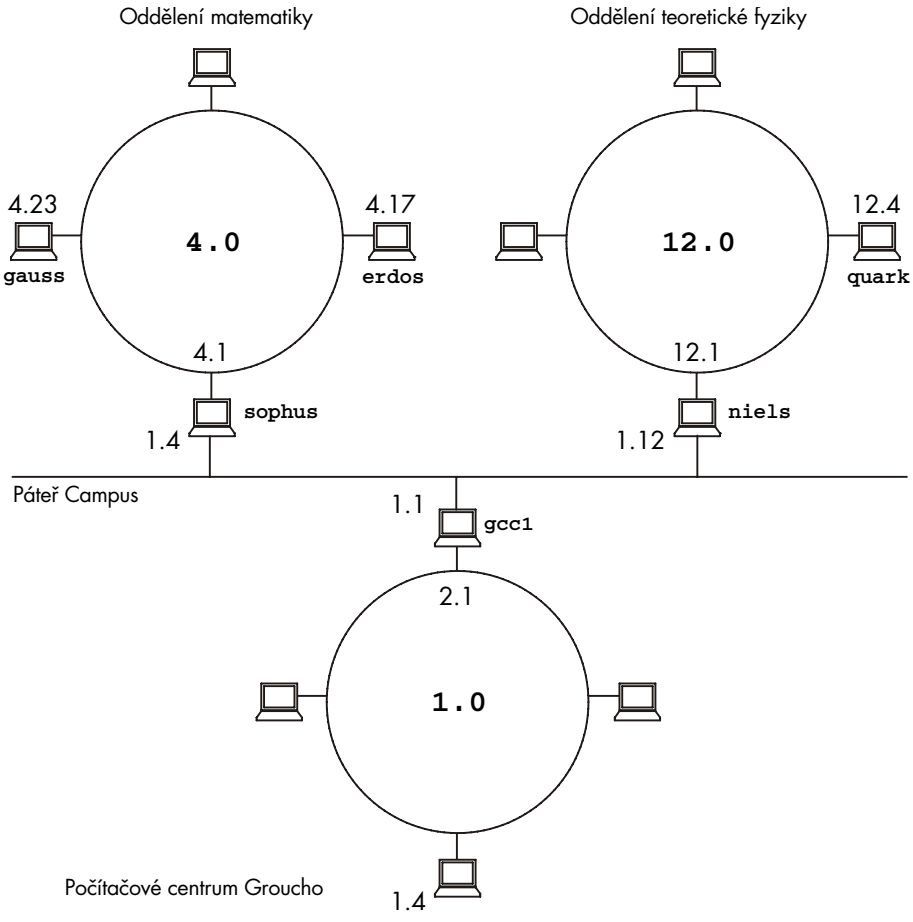
Obrázek 2.2 ukazuje část síťové topologie na Groucho Marx University (GMU). Hostitelé, kteří jsou v daném okamžiku ve dvou podsítích, jsou zobrazeni s oběma adresami.

Jemnou odlišnost mezi přidělením adresy hostiteli nebo jeho rozhraní je možné ignorovat. Na hostitele, kteří se nachází v jedné síti, například hostitel **erdos**, se budete většinou odkazovat pomocí IP adresy, i když třeba tato adresa odpovídá ethernetovému rozhraní. Tento rozdíl je ale důležitý pouze v případě, kdy se odkazujete na bránu.

2.4.4 Směrovací tabulka

Nyní se zaměříme na způsob, jakým protokol IP vybírá bránu, kterou použije k doručení datagramu do vzdálené sítě.

Ukázali jsme si, že u datagramu pro server **quark** zkontroluje hostitel **erdos** cílovou adresu, načež zjistí, že se nenachází v lokální síti. Proto ho pošle na implicitní bránu **sophus**, která je postavena před stejný úkol. Brána **sophus** pozná, že hostitel **quark** není na žádné ze sítí, s nimiž je přímo spojena, takže musí najít další bránu, na kterou by datagram poslala. Správnou volbou by byla brána ***niels***, což je brána katedry fyziky. Proto potřebuje brána **sophus** příslušné informace, aby mohla přiřadit cílovou síť vhodné bráně.



Obrázek 2.2
Část ze síťové topologie Groucho Marx University

Směrovací informace, které protokol IP pro tento účel používá, jsou tvořeny tabulkou, jež přiřazuje jednotlivé sítě branám, s nimiž jsou spojeny. Tabulka musí být doplněna o data typu zachyť-vše (catch all) (pro *implicitní směr*); je to brána sdružená se sítí **0.0.0.0**. Všechny pakety pro neznámou síť jsou poslány implicitním směrem. Pro bránu **sophus** by mohla tabulka vypadat následovně.

Síť	Brána	Rozhraní
149.76.1.0	**-**	**fddi0**
149.76.2.0	**149.76.1.2**	**fddi0**
149.76.3.0	**149.76.1.3**	**fddi0**
149.76.4.0	**-**	**eth0**
149.76.5.0	**149.76.1.5**	**fddi0**
…	**…**	**…**
0.0.0.0	**149.76.1.2**	**fddi0**

V sloupci brána je symbol „-“ proto, že se směruje do sítě, s níž je brána **sophus** přímo spojena, tudíž není žádná brána vyžadována.

Směrovací tabulky mohou být vytvářeny různým způsobem. U malých sítí LAN je většinou nejefektivnější vytvořit je ručně a při procesu zavádění je naplnit IP adresami pomocí příkazu `route` (viz. kapitola 5). V rozsáhlejších sítích jsou vytvářeny a přizpůsobovány v čase spuštění pomocí *směrovacích démonů*; démoni běží na centrálních hostitelích sítě a vyměňují si směrovací informace, čímž určí „optimální“ směrování mezi členskými sítěmi.

V závislosti na velikosti sítě budou použity různé směrovací protokoly. Ke směrování uvnitř autonomních systémů (jako je školní síť Groucho Marx) budou použity *vnitřní směrovací protokoly*. Nejvýznamnějším z nich je protokol RIP, což je směrovací informační protokol (Routing Information Protocol), který je implementován do *BSD-démona* `routed`. Ke směrování mezi autonomními systémy by měly být použity *externí směrovací protokoly*, jako je například EGP (vnější bránový protokol (External Gateway Protocol)) nebo BGP (hraniční bránový protokol (Border Gateway Protocol)); tyto protokoly (stejně tak i RIP) byly implementovány do *démona* `gated`.[3]

2.4.5 Metrické hodnoty

Dynamické směrování založené na protokolu RIP vybírá nejlepší směrování k cílovému hostiteli nebo síti na základě počtu „skoků“ (hops), to znamená počtu bran, kterými musí datagram projít, než dosáhne cíle. Čím kratší je směr, tím rychlejší je RIP. Velmi dlouhé směry s 16-ti nebo více skoky jsou považovány za nepoužitelné a jsou zrušeny.

[3] *Démona* `routed` považuje mnoho lidí za překonaný. Protože *démon* `gated` samozřejmě podporuje i protokol RIP, je lepší ho používat místo *démona* `routed`.

Chcete-li použít protokol RIP na vnitřní správu směrovacích informací ve vaší místní síti, musíte mít na všech hostitelích spuštěn program `gated`. Při procesu zavádění zkontroluje program `gated` všechna aktivní síťová rozhraní. Pokud existuje více než jedno aktivní rozhraní (nepočítaje zpětnovazebné rozhraní), bude předpokládat, že hostitelé posílají pakety mezi několika sítěmi a bude aktivně vyměňovat a vysílat směrovací informace. V opačném případě bude jen pasivně přijímat všechny aktualizace z protokolu RIP a aktualizovat místní směrovací tabulku.

Při vysílání informace z místní směrovací tabulky vypočte program `gated` délku směrování z tzv. *metrické hodnoty*, která je ve směrovací tabulce součástí dat jednotlivých položek. Tato metrická hodnota je při konfiguraci směrování nastavena systémovým správcem a měla by odrážet aktuální váhu použití daného směru. Proto by měla být metrická hodnota směrování k podsíti, se kterou je hostitel přímo propojen, vždy rovna nule, zatímco směr procházející dvěma bránami by měl mít metrickou hodnotu rovnu dvěma. S těmito metrickými hodnotami si však nemusíte dělat starosti, pokud nebudete používat protokol RIP nebo program `gated`.

2.5 Internetový protokol na řízení zpráv

Protokol IP obsahuje doprovodný protokol, o kterém jsme se zatím nezmínili. Jedná se o tzv. *internetový protokol na řízení zpráv* (*Internet Control Message Protocol*) (ICMP), který využívá síťový kód jádra operačního systému k předávání chybových zpráv ostatním hostitelům apod. Předpokládejme například, že jste znovu na hostiteli **erdos** a chcete se za pomoci `telnetu` připojit na port 12 345 serveru **quark**, ale na tomto portu neexistuje žádný odposlech. Pokud dorazí na tomto portu na server **quark** první paket, síťová hladina to pozná a okamžitě vrátí za pomoci protokolu ICMP hostiteli **erdos** zprávu, v níž bude uvedeno „Port je nedostupný“ (Port Unreachable).

Protokol ICMP rozumí poměrně velkému počtu zpráv, z nichž mnohé počítají s chybovými stavy. Mezi nimi existuje jedna velice zajímavá zpráva Přesměrování zprávy (Redirect message). Generuje je směrovací modul v případě, že zjistí, že ho již jako bránu používá jiný hostitel, i když ve skutečnosti existuje i mnohem kratší cesta. Například po restartu systému může být směrovací tabulka brány **sophus** neúplná, obsahujíc směrování na síť katedry matematiky, na páteř FDDI a implicitní směr na centrální bránu Groucho Computing Center (**gcc1**). Proto bude každý paket pro server **quark** poslán na bránu **gcc1** místo aby byl poslán přímo na bránu **niels**, což je brána katedry fyziky. Při přijetí takového datagramu si brána **gcc1** všimne, že jde o špatnou volbu, pošle paket na bránu **niels** a zároveň vrátí bráně **sophus** pomocí protokolu ICMP zprávu o přesměrování, která bude obsahovat výhodnější cestu.

Teď to vypadá, že jde o velmi chytrý způsob, jak se vyhnout manuálnímu nastavování všech směrů s výjimkou těch nejzákladnějších. Ale ne vždy je dobré spoléhat na dynamická směrovací schémata, jako jsou protokoly RIP nebo ICMP se zprávou o přesměrování. Zpráva o přesměrování u protokolu ICMP nebo protokol RIP nabízí jen malou, případně nulovou kontrolu toho, zda je směrovací informace skutečně autentická. Takto mohou zlomyslní uživatelé přerušit dopravu v celé síti nebo případně provést i něco horšího. Proto existují určité verze linuxového síťového kódu, které zacházejí se zprávami o přesměrování, jež ovlivňují směrování sítí, jakoby šlo jen o zprávy o přesměrování hostitelů.

2.6 Systém DNS

2.6.1 Rozlišení názvu hostitele

Jak jsme řekli, je adresování v sítích na bázi protokolu TCP/IP prováděno prostřednictvím 32bitových čísel. Pokud byste si však chtěli zapamatovat o něco více, strávili byste nad nimi poměrně hodně času. Proto se hostitelé označují „běžnými“ názvy, jako je hostitel **gauss** nebo hostitel **strange**. Povinností aplikace je, aby našla IP adresu odpovídající danému názvu. Tento proces se označuje jako *rozlišení názvu hostitele*.

Aplikace, která chce najít IP adresu příslušného názvu hostitele, nemusí obsahovat vlastní rutiny pro hledání hostitelů a jejich IP adres. Místo toho se spoléhá na skupinu funkcí z knihoven, které tuto operaci provádějí transparentně. Funkce se nazývají *gethostbyname(3)* a *gethostbyaddr(3)*. Obvykle jsou tyto funkce společně s mnoha dalšími příbuznými procedurami seskupeny v oddělené knihovně, v tzv. knihovně resolveru; v Linuxu jsou součástí standardní knihovny *libc*. Z toho důvodu se tato sbírka funkcí označuje jako tzv. „*resolver*.“

U malých sítí, jako je Ethernet, nebo dokonce jeho části, je správa tabulek s názvy hostitelů společně s jejich adresami příliš složitá. Tato informace je obvykle uchovávána v souboru s názvem `/etc/hosts`. Při přidávání nebo odebírání hostitele nebo opětovném přidělování adresy stačí na všech hostitelích aktualizovat soubor `hosts`. Tento proces se komplikuje u sítí, které jsou složeny z většího počtu počítačů.

Jedním z řešení tohoto problému je systém NIS, *Síťový informační systém* (*Network Information System*) vyvinutý firmou Sun Microsystems, hovorově se mu také říká YP, nebo-li *Žluté stránky* (*Yellow Pages*). Systém NIS ukládá soubor `hosts` (a další informace) do databáze na hlavním hostiteli, odkud ho mohou klienti dle potřeby získat. Přesto je tento přístup vhodný pouze pro středně velké sítě, jako jsou sítě LAN, protože vyžaduje centrální správu celé databáze `hosts` a její distribuci na všechny servery.

V Internetu byly adresy zprvu uchovávány v jediné databázi nazvané `HOSTS.TXT`. Tento soubor byl spravován síťovým informačním centrem (NIC) a všechny zúčastněné systémy si ho musely stáhnout a nainstalovat. Jak síť rostla, objevilo se v souvislosti s tímto schématem několik problémů. Kromě administrativního přetížení, které bylo způsobeno pravidelnou instalací souboru `HOSTS.TXT`, se příliš zvětšovalo i zatížení serverů, které ho distribuovaly. Mnohem horší ale bylo, že všechny názvy musely být registrovány v centru NIC. To proto, aby se některý název neobjevil dvakrát.

Proto došlo v roce 1984 k adaptaci nového schématu pro rozlišování názvů. Byl jím tzv. *Doménový jmenný systém* (*Domain Name System*). Systém DNS byl navržen Paulem Mockapetrisem a oba tyto problémy řeší současně.

2.6.2 Data DNS

Systém DNS organizuje názvy hostitelů do hierarchie domén. Doména je skupina systémů, mezi nimiž je nějaký vztah – buď proto, že tvoří vlastní síť (například všechny univerzitní počítače nebo všichni hostitelé v síti BITNET), nebo proto, že všechny patří určité organizaci (jako je například vláda Spojených států), nebo zkrátka tvoří geografické celky. Například univerzity jsou seskupeny do domény **edu**, každá univerzita nebo vysoké učení používá svoji *subdoménu*, pod kterou jsou sloučeni její hostitelé. Groucho Marx University má přidělenu doménu **groucho.edu** a síť LAN katedry matematiky má přidělenu subdoménu **maths.groucho.edu**. Hostitelé v síti katedry by měli tento název domény připojen ke svému názvu hostitele; takže hostitel **erdos** bude známý jako hostitel **erdos.maths.groucho.edu**. Tato symbolika je označována jako *plně kvalifikovaný název domény* (*fully qualified domain name*), nebo zkráceně FQDN, a jednoznačně identifikuje tohoto hostitele po celém světě.

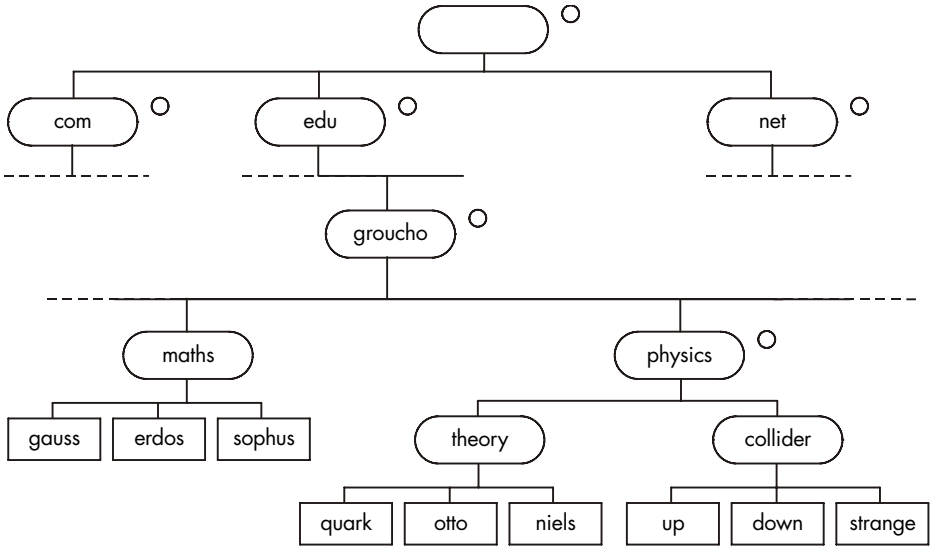
Obrázek 2.3 ukazuje část prostoru s názvy domén. Vstup u kořene tohoto stromu, který je označen malou tečkou, je poměrně výstižně nazván *kořenovou doménou* (*root domain*) a obklopuje všechny další domény. Aby se naznačilo, že název hostitele je plně kvalifikovaným názvem domény a nikoliv relativním názvem nějaké (implicitní) místní domény, píše se někdy s postfixovou tečkou. Ta udává, že název poslední části představuje kořenovou doménu.

V závislosti na jejím umístění v hierarchii názvů může být doména nazvána doménou nejvyšší úrovně, druhé úrovně nebo třetí úrovně. Více úrovní rozdělení sice je možných, ale vyskytují se jen řídce. Následuje několik domén nejvyšší úrovně, s nimiž se můžete často setkat:

edu Vzdělávací instituce (většinou v USA), jako jsou univerzity atd.

com Komerční organizace, společnosti.

org Nekomerční organizace. Často jsou v této doméně soukromé sítě typu UUCP.



Obrázek 2.3
Část prostoru s názvy domén

net Brány a další administrativní hostitelé na síti.

mil Vojenské instituce Spojených států amerických.

gov Vládní instituce Spojených států amerických.

uucp Oficiálně byly do této domény přesunuty názvy všech systémů, které byly dříve používány jako názvy UUCP bez domén.

Z technického hlediska patří první čtyři domény k části Internetu Spojených států amerických, ale i v těchto doménách můžete nalézt neamerické systémy. To platí zejména pro doménu **net**. Avšak domény **mil** a **gov** jsou používány výhradně Spojenými státy americkými.

Obecně platí, že každá země mimo území USA používá vlastní doménu nejvyšší úrovně, která je tvořena dvoupísmenným kódem země dle definice standardu ISO-3166. Například Finsko používá doménu **fi**, doména **fr** je přidělena Francii, doména **de** Německu, doména **au** patří Austrálii atd. Pod touto doménou nejvyšší úrovně může centrum NIC každé země libovolným způsobem organizovat názvy hostitelů. Například Austrálie má doménu druhé úrovně podobnou mezinárodním doménám nejvyšší úrovně, a sice **com.au**, **edu.au** atd. Ostatní země, jako je Německo, tuto speciální úroveň nepožívají, ale využívají raději delší názvy, které odkazují přímo na organizace, jež mají zvláštní doménu. Například není nic výjimečného setkat se s hostitelem, jehož název je například **ftp.informatik.uni-erlangen.de**. Zřejmě to nijak neovlivňuje německou výkonnost.

U těchto národních domén samozřejmě nemusí platit, že hostitel pod příslušnou doménou skutečně leží v dané zemi; pouze to signalizuje, že hostitel byl registrován centrem NIC dané země. Švédský výrobce může mít filiálku v Austrálii a přesto bude mít všechny své hostitele registrovány pod doménou nejvyšší úrovně **se**. Organizováním názvů do hierarchie názvů domén se tedy elegantně vyřeší problém jedinečnosti názvů; v systému DNS musí být název hostitele jedinečný pouze uvnitř vlastní domény, protože ta mu dává název odlišný od ostatních hostitelů po celém světě. Kromě toho jsou plně kvalifikované názvy poměrně lehce zapamatovatelné. Pak je velmi vhodné rozdělit rozsáhlou doménu na několik subdomén.

Ale systém DNS toho umí ještě mnohem více: umožňuje pověřit správou subdomény její správce. Například správci v instituci Groucho Computing Center mohou vytvořit subdoménu pro každý ústav. Již jsme se setkali se subdoménami **maths** a **physics**. Pokud se bude zdát správcům síť katedry fyziky pro správu zvenčí příliš rozsáhlá a chaotická (konec konců fyzici jsou známí jako neovladatelná skupina lidí), mohou jednoduše předat řízení nad doménou **physics.groucho.edu** správcům této sítě. Ti potom mohou používat libovolné názvy hostitelů a přidělovat jim IP-adresy své sítě, aniž by do toho zvenčí někdo zasahoval.

Na konci řetězce je prostor s názvy rozdělen na *zóny* (*zones*), z nichž každá je směrována na doménu. Všimněte si jemné odlišnosti mezi zónou a doménou: *doména* **groucho.edu** obsahuje všechny hostitele instituce Groucho Marx University, zatímco *zóna* **groucho.edu** obsahuje pouze hostitele, které přímo spravuje centrum Computing Center, například hostitele z katedry matematiky. Hostitelé z katedry fyziky patří do odlišné zóny, konkrétně **physics.groucho.edu**. Na obrázku 2.3 je začátek zóny označen malým kolečkem napravo od názvu domény.

2.6.3 Vyhledávání názvů s pomocí DNS

Na první pohled se může zdát, že u všech těchto rozsáhlých domén a zón je provedení rozlišení názvu nesmírně komplikované. Konec konců, neřídí-li názvy, které jsou přidělovány daným hostitelům, žádná centrální správa, jak je má potom aplikace na nižší úrovni uhodnout?

Nyní přichází na řadu bezelstná vlastnost DNS. Chcete-li nalézt IP-adresu serveru **erdos**, pak vám DNS odpoví, že se máte jít zeptat lidí, kteří ho spravují, a oni vám ji řeknou.

Systém DNS je de facto obrovskou distribuovanou databází. Je implementován za pomoci takzvaných jmenných serverů (name servers), které dodávají informace o dané doméně nebo o dané skupině domén. U každé zóny existují nejméně dva a nejvýše několik jmenných serverů, které uchovávají všechny závažné informace o hostitelích v příslušné zóně. Pokud chcete získat IP-adresu serveru **erdos**, pak se stačí spojit s jmenným serverem zóny **groucho.edu**, který vám následně vrátí požadovaná data.

Možná si myslíte, že se o tom mnohem snadněji mluví, ale hůře se to provádí. Takže, jak mám vědět, jak se spojit s jmenným serverem na Groucho Marx University? V případě, že váš počítač není vybaven nástrojem pro analýzu adres, provede to systém DNS za něj. Když chce vaše aplikace vyhledat informace o serveru **erdos**, spojí se s místním jmenným serverem, který aplikuje na dané informace tzv. iterační dotaz. Začne zasláním dotazu jmennému serveru v kořenové doméně, kde se zeptá na adresu **erdos.maths.groucho.edu**. Kořenový jmenný server pozná, že tento název nepatří do jeho správní zóny, ale patří do zóny pod doménou **edu**. Takže vám sdělí, že se máte spojit s jmenným serverem zóny **edu**, kde získáte více informací. Ke své odpovědi dále přibalí i seznam všech jmenných serverů domény **edu** společně s jejich adresami. Potom bude váš místní jmenný server pokračovat a pošle dotaz na některý z nich, například na **a.isi.edu**. Podobným způsobem jako u kořenového jmenného serveru se server **a.isi.edu** dozví, že lidé z **groucho.edu** mají svou vlastní zónu a odkáže vás na její vlastní servery. Místní jmenný server bude pokračovat v dotazu na server **erdos** na jednom z těchto serverů, který konečně zpozná, že daný server patří do jeho zóny, a vrátí jeho odpovídající IP-adresu.

Teď to možná vypadá, že kvůli vyhledání jedné mizerné IP-adresy bylo zapotřebí provést spoustu operací, ale ve skutečnosti je to v porovnání s množstvím dat, které by musely být přeneseny, kdyby se stále pracovalo se souborem `HOSTS.TXT`, minimum. Stále však existuje prostor, jak toto schéma vylepšit.

Aby zkrátil dobu odpovědi při dalších dotazech, uchovává jmenný server získané informace ve své místní *vyrovnávací paměti (cache)*. Takže když chce příště někdo z vaší lokální sítě vyhledat adresu hostitele v doméně **groucho.edu**, nemusí jmenný server znovu absolvovat celý výše popsaný proces, ale přímo se spojí s jmenným serverem pro doménu **groucho.edu**.[4]

Samozřejmě, že server nebude tyto informace uchovávat donekonečna, ale po určité době je smaže. Tato doba platnosti se nazývá *čas přežití* (*time to live*), zkráceně TTL. V databázi DNS přiděluje každé položce TTL správce, který je zodpovědný za danou zónu.

2.6.4 Servery systému DNS

Jmenné servery, které uchovávají všechny informace o hostitelích příslušné zóny, se nazývají *autoritativními jmennými servery* (*authoritative*) dané zóny a někdy jsou také označovány jako *hlavní jmenné servery* (*master name servers*). Jakýkoliv dotaz na hostitele uvnitř této zóny nakonec stejně skončí až u těchto hlavních jmenných serverů.

4 Pokud tak neučiní, je systém DNS stejně špatný, jako ostatní metody, protože každý dotaz bude vyžadovat zapojení kořenových jmenných serverů.

Aby byl zachován spojitý obraz zóny, musí být její hlavní servery poměrně dobře synchronizovány. Toho dosáhneme tak, že jednoho ze serverů označíme jako *primární* (*primary*). Ten bude načítat informace o své zóně z datových souborů a ostatní servery budou označeny jako *sekundární* (*secondary*) a budou v pravidelných intervalech přenášet data z primárního serveru.

Jedním z důvodů, proč bychom měli mít několik jmenných serverů, je rozložení pracovní zátěže, dalším důvodem je pak redundance. Když přestane některý z jmenných serverů správně pracovat, například když spadne nebo ztratí síťové spojení, přejdou všechny dotazy na ostatní servery. Samozřejmě vás toto schéma neochrání před chybami serveru, jejichž důsledkem budou špatné odpovědi na všechny požadavky systému DNS, například z důvodu softwarových chyb ve vlastním programu serveru.

Samozřejmě můžete chtít provozovat i takový jmenný server, který nebude správcem žádné domény[5]. Přesto však je tento typ serveru důležitý, protože je stále schopen provádět DNS-dotazy pro aplikace, které jsou spuštěny v lokální síti, a dále může ukládat informace do vyrovnávací paměti. Proto se tento typ serveru nazývá server *pouze pro cachování* (*caching-only* server).

2.6.5 Databáze systému DNS

Ukázali jsme si, že systém DNS nejenom určuje IP-adresy hostitelů, ale také vyměňuje informace mezi jmennými servery. Ve skutečnosti může databáze DNS obsahovat celý svazek různých typů záznamů.

V databázi DNS se jednotlivým informacím říká *zdrojový záznam* (*resource record*), zkráceně RR. Každý záznam má přidělený typ, který popisuje druh dat, jež reprezentuje a třídu, určující typ sítě, na kterou se bude záznam vztahovat. Třída uspokojuje potřeby různých adresních schémat, jako jsou IP-adresy (třída IN) nebo adresy sítí Hesiod (používaných v MIT) a některé další. Vzorovým typem zdrojového záznamu je záznam A, který sdružuje plně kvalifikované názvy domény s IP-adresou. Samozřejmě, že hostitel může mít více než jeden název. Nicméně jeden z těchto názvů musí být označen jako oficiální, neboli *kanonický název hostitele* (*canonical host name*) a ostatní jsou chápány jako přezdívky oficiálního názvu. Rozdíl spočívá v tom, že kanonický název hostitele je takový, se kterým je sdružen záznam typu A, zatímco ostatní názvy mají pouze záznam typu CNAME, který odkazuje na kanonický název hostitele.

Nebudeme zde procházet všechny typy záznamů, to si schováme do příští kapitoly. Nyní vám raději poskytneme krátký příklad. Obrázek 2.4 ukazuje část databázové domény, kterou mají načtenu jmenné servery zóny **physics.groucho.edu**.

5 Většinou je to v pořádku. Jmenný server by měl přinejmenším poskytovat jmenné služby pro **místního hostitele** a vracet vyhledávání adresy **127.0.0.1**.

```
;
; Autoritativní informace o doméně physics.groucho.edu
@                        IN     SOA            {
                        niels.physics.groucho.edu.
                        hostmaster.niels.physics.groucho.edu.
                        1034               ; serial no
                        360000             ; refresh
                        3600               ; retry
                        3600000            ; expire
                        3600               ; default ttl
                      }
;
; Jmenné servery
                            IN     NS         niels
                            IN     NS         gauss.maths.groucho.edu.
gauss.maths.groucho.edu.    IN A              149.76.4.23
;
; Teoretická fyzika (podsíť 12)
niels                       IN     A          149.76.12.1
                            IN     A          149.76.1.12
nameserver                  IN     CNAME      niels
otto                        IN     A          149.76.12.2
quark                       IN     A          149.76.12.4
down                        IN     A          149.76.12.5
strange                     IN     A          149.76.12.6
...
; Collider Lab. (podsíť 14)
boson                       IN     A          149.76.14.1
muon                        IN     A          149.76.14.7
bogon                       IN     A          149.76.14.12
...
```

Obrázek 2.4
Výpis ze souboru `named.hosts` pro katedru fyziky

Kromě záznamů typů A a CNAME můžete v horní části souboru vidět speciální záznam, který je roztažen na několika řádcích. Je to zdrojový záznam SOA, což označuje *začátek správy* (*Start of Authority*), kde jsou umístěny všeobecné informace o zóně, kterou daný server spravuje. Je v něm například obsažen implicitní čas přežití pro všechny záznamy.

Všimněte si, že všechny názvy v ukázkovém souboru, které nekončí tečkou, by měly být interpretovány vzhledem k doméně **groucho.edu**. Speciální název „@“ použitý v záznamu typu SOA odkazuje na vlastní název domény.

Řekli jsme si, že jmenné servery domény **groucho.edu** musí nějakým způsobem vědět o zóně **physics**, aby mohly odkazovat dotazy na její jmenné servery. Toho se obvykle dosáhne pomocí dvojice záznamů: záznam typu NS, který poskytne FQDN název serveru, a záznam typu A, který tomuto názvu přidruží adresu. Protože právě tyto záznamy drží jmenný prostor pohromadě, označují se často jako tzv. *tmelící záznamy* (*glue records*). Jsou jedinými případy záznamů, kde rodičovská zóna skutečně uchovává informace o hostitelích podřazené zóny. Na obrázku 2.5 ukazují tmelící záznamy na jmenné servery **physics.groucho.edu**.


```
;
; Data zóny groucho.edu.
@                    IN        SOA           {
                      vax12.gcc.groucho.edu.
                      hostmaster.vax12.gcc.groucho.edu.
                      233               ; serial no
                      360000            ; refresh
                      3600              ; retry
                      3600000           ; expire
                      3600              ; default ttl
                     }
....
;
; Tmelící záznamy pro doménu physics.groucho.edu
physics              IN      NS         niels.physics.groucho.edu.
                     IN      NS         gauss.maths.groucho.edu.
niels.physics        IN      A          149.76.12.1
gauss.maths          IN      A          149.76.4.23
...
```

Obrázek 2.5
Výpis ze souboru `named.hosts` pro GMU

2.6.6 Zpětné vyhledávání

Kromě vyhledávání IP-adresy, která náleží hostiteli, je někdy žádoucí vyhledat také název kanonického hostitele, který odpovídá příslušné adrese. Tento proces se označuje jako tzv. *zpětné mapování* (*reverse mapping*) a některé síťové služby ho používají k ověřování identity klienta. Pokud se používá jediný soubor `hosts`, pak si zpětné vyhledávání vyžádá vyhledání souboru hostitele, který vlastní požadovanou IP-adresu. U DNS samozřejmě nepřipadá v úvahu úplné prohledání jmenného prostoru. Místo toho byla vytvořena speciální doména **in-addr.arpa**, která obsahuje všechny IP-adresy všech hostitelů v převrácené tečkové notaci. Například IP-adrese **149.76.12.4** odpovídá název **4.12.76.149.in-addr.arpa**. Záznam typu PTR je takový záznam, který tyto názvy sdružuje s názvy svých kanonických hostitelů.

Vytvoření správní zóny obvykle znamená, že její správci mají plnou kontrolu nad způsobem, jakým jsou adresy přiřazovány k názvům. Protože většinou obsluhují jednu nebo více sítí nebo podsítí IP, existuje mezi DNS-zónami a IP-sítěmi jedno či více mapování. Například katedra fyziky obsahuje podsítě **149.76.8.0**, **149.76.12.0** a **149.76.14.0**.

V důsledku toho musí být nové zóny v doméně **in-addr.arpa** vytvářeny společně se zónou **physics** a musí být postoupeny síťovým správcům kateder: **8.76.149.in-addr.arpa**, **12.76.149.in-addr.arpa** a **14.76.149.in-addr.arpa**. Jinak by instalace nového hostitele v Collider Lab vyžadovala, aby se správci spojili se svou nadřazenou doménou, kde se nová adresa zapíše do souboru zóny **in-addr.arpa**.

Na obrázku 2.6 je zobrazena databáze zóny pro podsíť 12. Odpovídající tmelící záznamy v databázi jejich rodičovské zóny vidíte na obrázku 2.7.

```
;
; doména 12.76.149.in-addr.arpa.
@                   IN      SOA    {
                    niels.physics.groucho.edu.
                    hostmaster.niels.physics.groucho.edu.
                    233 360000 3600 3600000 3600
                  }
2                   IN      PTR        otto.physics.groucho.edu.
4                   IN      PTR        quark.physics.groucho.edu.
5                   IN      PTR        down.physics.groucho.edu.
6                   IN      PTR        strange.physics.groucho.edu.
```

Obrázek 2.6
Výtah ze souboru `named.rev` pro podsíť 12

V důsledku toho mohou být zóny vytvářeny pouze jako nadmnožiny sítí IP-a co je ještě krutější, musí mít síťové masky na hranicích bajtu. Všechny podsítě v Groucho Marx University mají síťovou masku **255.255.255.0**, takže lze pro každou podsíť vytvořit zónu **in-addr.arpa**. Pokud by však existovala síťová maska **255.255.255.128**, nebylo by možné vytvořit zónu pro podsíť **149.76.12.128**, protože by neexistoval žádný způsob, jak sdělit systému DNS, že doména **12.76.149.in-addr.arpa** byla rozdělena na dvě zóny se samostatnou správou a s názvy hostitelů od **1** do **127**, resp. od **128** do **255**.

```
;
; the 76.149.in-addr.arpa domain.
@                   IN        SOA           {
                    vax12.gcc.groucho.edu.
                    hostmaster.vax12.gcc.groucho.edu.
                    233 360000 3600 3600000 3600
                    }
...
; podsíť 4: Katedra matematiky
1.4                 IN      PTR       sophus.maths.groucho.edu.
17.4                IN      PTR       erdos.maths.groucho.edu.
23.4                IN      PTR       gauss.maths.groucho.edu.
...
; podsíť 12: Katedra fyziky
12                  IN      NS        niels.physics.groucho.edu.
                    IN      NS        gauss.maths.groucho.edu.
niels.physics.groucho.edu. IN  A 149.76.12.1
gauss.maths.groucho.edu. IN  A   149.76.4.23
...
```

Obrázek 2.7
Výtah ze souboru `named.rev` pro síť **149.76**

Kapitola

3

Konfigurace síťového hardwaru

3.1 Zařízení, ovladače atd.

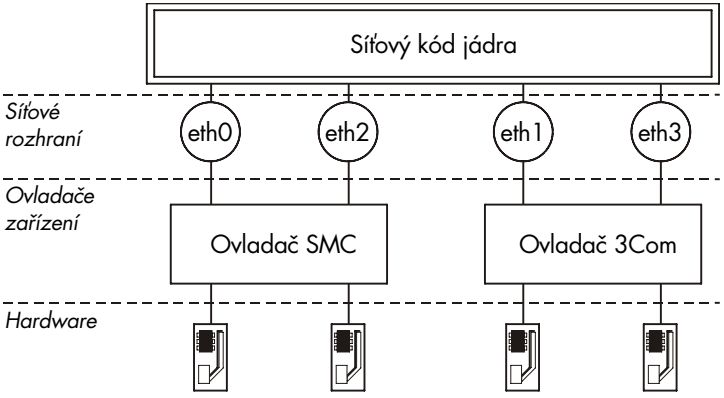
Až dosud jsme se bavili o síťových rozhraních a o všeobecných problémech protokolu TCP/IP, ale zatím jsme si neřekli, *co* se doopravdy děje, když „síťový kód“ jádra přistupuje k části hardwaru. Nyní si řekneme něco o koncepci rozhraní a ovladačů.

Nejprve máte samozřejmě vlastní hardware, například ethernetovou kartu: to je laminátová deska přecpaná spoustou mrňavých čipů, na kterých lze najít nějaká čísílka. Toto vše je umístěno ve slotu vašeho počítače a všeobecně se tomu říká zařízení.

Abyste vůbec mohli používat ethernetovou kartu, musí být v jádru Linuxu k dispozici speciální funkce, které rozumí způsobu, jakým je k tomuto zařízení přistupováno. Takovýmto funkcím říkáme ovladače zařízení. Například Linux obsahuje ovladače zařízení pro několik skupin ethernetových karet, které si jsou z funkčního hlediska hodně podobné. Jsou nazvány „Becker Series Drivers“ po svém autorovi Donaldu Beckerovi. Jiným příkladem je ovladač D-Link, který obsluhuje kapesní adaptér D-Link, jenž je připojen k paralelnímu portu.

Ale co myslíme tím, když říkáme, že ovladač se „stará“ o zařízení? Vraťme se zpět k ethernetové kartě, kterou jsme si popsali na začátku. Ovladač musí být nějakým způsobem schopen komunikovat s logikou hardwarové karty: musí jí posílat příkazy a data, a ta by mu měla na oplátku doručit všechna přijatá data.

U počítačů PC se tato komunikace odehrává v oblasti V/V paměti, která je namapována na registry karty. Všechny příkazy a data vyslané jádrem do karty musí těmito registry projít. V/V paměť je obecně určena zadáním počáteční, neboli *základní adresy* (*base address*). Typické základní adresy pro ethernetové karty jsou **0x300** nebo **0x360**.



Obrázek 3.1
Vztahy mezi ovladači, rozhraními a hardwarem

Obvykle není třeba si dělat příliš starosti s konfigurací hardwaru, jako je základní adresa, protože jádro se při zavádění pokusí detekovat pozici karty. Tento proces se nazývá automatické detekce, což znamená, že jádro přečte několik adresových míst a porovná přečtená data s tím, co by získal, kdyby na nich byla nainstalována určitá ethernetová karta. Mohou však existovat ethernetové karty, které automatickou detekci neumí; to bývá případ levných ethernetových karet, které nejsou dokonalými klony standardních karet jiných výrobců. Jádro systému také bude při zavádění zkoušet detekovat pouze jediné ethernetové zařízení. Pokud používáte více než jednu kartu, musíte jádru o této kartě explicitně říci.

Dalším takovým parametrem, který musíte jádru předat, je požadavek kanálu přerušení (interrupt request channel). Pokud potřebují hardwarové komponenty něco zařídit, obvykle předají jádru přerušení. Příkladem může býti obdržení dat nebo výskyt speciálních událostí. U počítačů PC se mohou přerušení vyskytnout na jednom z 15 kanálů přerušení, které jsou očíslovány 0, 1 a 3 až 15. Číslo přerušení, které je hardwarové komponentě přiděleno, se nazývá *číslo požadavku přerušení* (*interrupt request number*), zkráceně IRQ.[1]

V kapitole 2 jsme si řekli, že jádro přistupuje k zařízení pomocí tzv. rozhraní. Ta poskytují abstraktní množinu funkcí, jako je například posílání nebo příjem datagramu, které jsou pro všechny typy hardwaru totožné.

Rozhraní jsou identifikována prostřednictvím svého názvu. Tyto názvy jsou interně definovány v jádru, nejedná se však o soubory zařízení v adresáři `/dev`. Typickými názvy ethernetových rozhraní jsou *eth0*, *eth1* atd. Přiřazení rozhraní k jednotlivým zařízením obvykle závisí

[1] IRQ 2 a 9 jsou totožné, protože PC obsahuje dva procesory přerušení seřazené za sebou, z nichž každý má osm IRQ; druhý procesor je připojen na IRQ 2 prvního procesoru.

na pořadí, ve kterém jsou zařízení nakonfigurována; například první nainstalovaná ethernetová karta obdrží název *eth0*, další bude *eth1* atd. Jedinou výjimku z tohoto pravidla tvoří rozhraní SLIP, která jsou přidělována dynamicky; to znamená, kdykoliv je uskutečněno spojení SLIP, je rozhraní přiřazeno sériovému portu.

Schéma uvedené na obrázku 3.1 se pokouší ukázat vztahy mezi hardwarem, ovladači zařízení a rozhraními. Při zavádění zobrazí jádro detekovaná zařízení a rozhraní, která se mu podařilo nainstalovat. Následuje výpis typické obrazovky při zavádění systému:

```
.
.
This processor honours the WP bit even when in supervisor mode. Good.
Floppy drive(s): fd0 is 1.44M
Swansea University Computer Society NET3.010
IP Protocols: ICMP, UDP, TCP
PPP: version 0.2.1 (4 channels) OPTIMIZE_FLAGS
TCP compression code copyright 1989 Regents of the University
  of California
PPP line discipline registered.
SLIP: version 0.7.5 (4 channels)
CSLIP: code copyright 1989 Regents of the University of California
dl0: D-Link DE-600 pocket adapter, Ethernet Address: 00:80:C8:71:76:95
Checking 386/387 coupling...OK, fpu using exception 16 error reporting.
Linux version 1.1.11 (okir@monad) #3 Sat May 7 14:57:18 MET DST 1994
```

Výpis ukazuje, že jádro bylo zkompilováno se zapnutým protokolem TCP/IP, včetně ovladačů pro protokoly SLIP, CSLIP a PPP. Třetí řádka odspodu uvádí, že byl detekován kapesní adaptér D-Link a nainstalován jako rozhraní *dl0*. Máte-li jiný typ ethernetové karty, vypíše jádro zpravidla řádek začínající *eth0*, za nímž následuje typ detekované karty. Pokud máte nainstalovanou ethernetovou kartu, ale nevidíte žádnou takovou zprávu, znamená to, že jádro není schopno správně detekovat vaši kartu. Tímto problémem se budeme zabývat v další části.*

* Poznámka korektora: V současné době jsou podporována i tzv. modulární jádra, kdy kromě vlastního jádra existují ještě moduly, které mohou být v průběhu činnosti jádra „nataženy“ a používány stejně jako ovladače přímo v jádru.

3.2 Konfigurace kernelu (jádra)

Většina distributorů Linuxu dodává zaváděcí diskety, které pracují se všemi běžnými typy hardwaru počítačů PC. To znamená, že jádro na těchto disketách je zkompilováno se všemi skupinami ovladačů, které však nikdy nebudete potřebovat všechny a které plýtvají drahocennou pamětí, protože části jádra nelze z paměti zrušit. Proto si obvykle sestavíte své vlastní jádro, jež bude obsahovat pouze ovladače, které skutečně chcete nebo potřebujete.

Chcete-li používat systém Linux, měli byste ovládat sestavování jádra. Základy jsou vysvětleny v manuálu Matta Welshe „Installation and Getting Started“, který je součástí série Linux – dokumentační projekt. Proto se v této stati zmíníme pouze o těch konfiguračních volbách, které ovlivňují nastavení sítí.

Při spuštění příkazu `make config` budete nejprve dotazováni na všeobecné konfigurace, například zda chcete či nechcete jádro s emulací matematického koprocesoru atd. Jeden z těchto dotazů se bude týkat podpory pro sítě na bázi TCP/IP. Aby bylo vaše jádro schopno pracovat se sítěmi, musíte na tento dotaz odpovědět **y** (yes - ano).

3.2.1 Volby jádra Linuxu verze 1.0 a vyšší

Jakmile je dokončena obecná část voleb, bude konfigurace pokračovat dotazy na různé rysy systému, jako jsou ovladače SCSI apod. Následující seznam otázek se pak bude týkat podpory sítí. Přesná množina konfiguračních voleb se z důvodu probíhajícího vývoje stále mění. Typický seznam voleb, který nabízí většina verzí jádra mezi 1.0 a 1.1, vypadá následovně:

```
*
* Network device support
*
Network device support? (CONFIG_ETHERCARDS) [y]
```

Pokud chcete používat *nějaký* typ síťových zařízení, ať se už jedná o protokol Ethernet, SLIP nebo PPP, musíte na tuto otázku odpovědět **y** (bez ohledu na název makra, který je uveden v kulatých závorkách). Pokud na tuto otázku odpovíte kladně, bude automaticky povolena podpora ethernetových typů zařízení. Podpora ostatních typů síťových ovladačů musí být povolena samostatně:

```
SLIP (serial line) support? (CONFIG_SLIP) [y]
SLIP compressed headers (SL_COMPRESSED) [y]
PPP (point-to-point) support (CONFIG_PPP) [y]
PLIP (parallel port) support (CONFIG_PLIP) [n]
```

Tyto otázky se týkají spojení pomocí různých protokolů, které Linux podporuje. Protokol SLIP vám umožní posílat IP-datagramy po sériové lince. Možnost komprimovat hlavičku poskytuje podporu pro protokol CSLIP. Tato technika zkomprimuje hlavičky TCP/IP až na tři bajty. Všimněte si, že tato volba jádra nezapíná automaticky protokol CSLIP, pouze mu zprostředkovává nezbytné funkce jádra.

PPP je dalším protokolem, který umožňuje posílání síťových dat po sériové lince. Je mnohem pružnější, než protokol SLIP a není omezen jen na protokol IP, ale podporuje i protokol IPX, je-li tento nainstalován.

PLIP nabízí způsob, jak posílat IP-datagramy za pomoci spojení přes paralelní port. Většinou se používá ke komunikaci s počítači PC, na kterých běží operační systém DOS.

Následující otázky se budou zabývat ethernetovými kartami od různých výrobců. Protože se stále vyvíjí nové ovladače, může být tato část rozšířena o další otázky. Pokud chcete sestavit jádro, které budete používat na více strojích, měli byste povolit více než jeden ovladač.

```
NE2000/NE1000 support (CONFIG_NE2000) [y]
WD80*3 support (CONFIG_WD80x3) [n]
SMC Ultra support (CONFIG_ULTRA) [n]
3c501 support (CONFIG_EL1) [n]
3c503 support (CONFIG_EL2) [n]
3c509/3c579 support (CONFIG_EL3) [n]
HP PCLAN support (CONFIG_HPLAN) [n]
AT1500 and NE2100 (LANCE and PCnet-ISA) support (CONFIG_LANCE) [n]
AT1700 support (CONFIG_AT1700) [n]
DEPCA support (CONFIG_DEPCA) [n]
D-Link DE600 pocket adaptor support (CONFIG_DE600) [y]
AT-LAN-TEC/RealTek pocket adaptor support (CONFIG_ATP) [n]
*
* CD-ROM drivers
*
...
```

Nakonec se vás konfigurační script v části věnované systémům souborů zeptá, zda chcete podporu pro NFS, síťový souborový systém (network filesystem). NFS umožňuje exportovat souborové systémy na několik hostitelů, takže se vám bude zdát, že jsou soubory uloženy na běžném pevném disku, který je připojen k danému hostiteli.

```
NFS filesystem support (CONFIG_NFS_FS) [y]
```

3.2.2 Volby jádra Linuxu verze 1.1.14 a vyšší

Od verze 1.1.14, do které byla přidána alfa-verze protokolu IPX, se lehce změnila konfigurační procedura. Část s obecnými volbami se nyní zeptá, zda si přejete nějakou podporu sítí. Pak ihned následuje skupina dotazů týkající se síťových voleb.

```
*
* Networking options
*
TCP/IP networking (CONFIG_INET) [y]
```

Aby bylo možné používat sítě na bázi TCP/IP, musíte na tuto otázku odpovědět **y**. Pokud však odpovíte **n**, i nadále bude možné sestavit jádro s podporou protokolu IPX.

```
IP forwarding/gatewaying (CONFIG_IP_FORWARD) [n]
```

Tuto volbu byste měli povolit v případě, že váš systém bude vystupovat jako brána mezi dvěma Ethernety, nebo mezi Ethernetem a spojením SLIP atd. I když příliš nevadí, když tuto volbu implicitně povolíte, budete možná chtít v případě konfigurace hostitele jako tzv. firewall tuto volbu zakázat. Firewally jsou hostitelé, kteří jsou připojeni ke dvěma nebo více sítím, avšak nesměrují mezi nimi síťový provoz. Obecně poskytují uživatelům firemní sítě přístup k Internetu, a to s minimálními riziky pro vnitřní síť. Uživatelé se budou moci přihlašovat k firewallu a používat internetové služby, ale počítače ve firmě budou chráněny před vnějšími útoky, protože žádné příchozí spojení nemůže přes firewall přejít.

```
*
* (it is safe to leave these untouched)
*
PC/TCP compatibility mode (CONFIG_INET_PCTCP) [n]
```

Za pomoci této volby zabráníte nekompatibilitě s některými verzemi PC/TCP, což je komerční implementace TCP/IP u počítačů PC s operačním systémem DOS. Pokud tuto volbu povolíte, budete stále moci komunikovat s běžnými unixovými stroji, avšak u pomalých linek může dojít ke ztrátě výkonu.

```
Reverse ARP (CONFIG_INET_RARP) [n]
```

Tato funkce povolí RARP, protokol pro zpětné rozlišení adres. Protokol RARP je používán u bezdiskových klientů a X-terminálů, kde při zavádění slouží ke zjištění IP-adresy. Protokol RARP byste měli povolit jen v případě, že hodláte provozovat tento typ klientů. Poslední balík síťových utilit (*net-0.32d*) obsahuje malou utilitu s názvem *rarp*, která umožňuje přidávat systémy do vyrovnávací paměti RARP.

```
Assume subnets are local (CONFIG_INET_SNARL) [y]
```

Posíláte-li data přes TCP, musí je jádro před předáním protokolu IP rozdělit do několika IP-paketů. U hostitelů, kteří jsou dosažitelní po místní síti, jako například Ethernet, se použijí větší pakety než u hostitelů, pro jejichž dosažení musí urazit dlouhou cestu.[2] Pokud nepovolíte *SNARL*, bude jádro předpokládat, že lokální sítě jsou pouze takové, se kterými má skutečné rozhraní. Pokud se však podíváte na síť třídy B v Groucho Marx University, uvidíte, že celá síť třídy B je lokální, avšak většina hostitelů má rozhraní pouze s jednou nebo se dvěma podsítěmi. Jestliže *SNARL* povolíte, bude jádro předpokládat, že *všechny* podsítě jsou lokální a pro komunikaci se všemi hostiteli v rámci univerzity použije velké pakety.

Pokud chcete pro data, která jsou posílána konkrétním hostitelům, používat pakety o malé velikosti (protože například data putují po spojení SLIP), můžete tak učinit za pomoci volby *mtu* příkazu *route*; ta je stručně probrána na konci této kapitoly.

```
Disable NAGLE algorithm (normally enabled) (CONFIG_TCP_NAGLE_OFF) [n]
```

Pravidlo NAGLE je heuristické a zabraňuje posílání zvláště malých IP-paketů, někdy nazývaných minigramy (tinygrams). Minigramy jsou obvykle tvořeny nástroji pro interaktivní práci se sítí, které přenášejí jednotlivé stisky kláves, jako například *telnet* nebo *rsh*. Minigramy jsou však mimořádně nehospodárné u spojení s malou šířkou pásma, například pomocí protokolu SLIP. Algoritmus NAGLE se tomu může pokusit zabránit tak, že za určitých okolností na chvíli pozdrží vyslání dat za pomoci protokolu TCP. Máte-li vážné problémy se ztracenými pakety, pak můžete algoritmus NAGLE zakázat.

```
The IPX protocol (CONFIG_IPX) [n]
```

Tato volba povolí protokol IPX, což je transportní protokol, který používají sítě Novell. Jednou z výhod může být možnost výměny dat s utilitami DOS, které využívají protokol IPX, nebo směrování dopravy mezi vašimi sítěmi Novell prostřednictvím spojení PPP.

Od verze jádra 1.1.16 podporuje Linux ještě jeden typ ovladače, tzv. fiktivní ovladač. Následující otázka se týká počátku sekce, ve které se nastavují ovladače zařízení.

```
Dummy net driver support (CONFIG_DUMMY) [y]
```

Fiktivní ovladač toho příliš mnoho neumí, ale je poměrně užitečný u samostatných hostitelů nebo hostitelů SLIP. V podstatě jde o maskující zpětnovazebné rozhraní. Důvodem pro tento druh rozhraní, jsou hostitelé, kteří poskytují SLIP, ale nemají žádný Ethernet. Přesto však potřebujete rozhraní, které bude mít po celou dobu přiřazenu IP-adresu. Podrobněji je tento problém probírán ve stati Fiktivní rozhraní v kapitole 5.

[2] To z důvodu, abychom se vyhnuli fragmentaci u těch spojení, u kterých lze nastavit maximální velikost paketu na velmi malou hodnotu.

3.3 Průvodce linuxovými síťovými zařízeními

Jádro Linuxu podporuje množství ovladačů hardwaru pro různé typy zařízení. Tato stať obsahuje krátký přehled dostupných rodin ovladačů a názvů rozhraní, které ovladače používají.

V Linuxu existuje množství standardních názvů rozhraní, jejichž výčet následuje. Většina ovladačů podporuje více než jedno rozhraní. V takovém případě jsou rozhraní číslována,např. *eth0*, *eth1* atd.

lo Lokální zpětnovazebné rozhraní. Je používáno za účelem testování, stejně jako dvojice síťových aplikací. Pracuje jako uzavřený obvod, kde všechny datagramy, které jsou na něj poslány, jsou okamžitě. V jádru existuje vždy jedno zpětnovazebné rozhraní. Menší nebo větší počet těchto rozhraní nemá smysl.

ethn Ethernetová karta číslo *n*. To je obecný název rozhraní pro většinu ethernetových karet.

dln Tato rozhraní přistupují ke kapesnímu adaptéru D-Link DE-600, což je další ethernetové zařízení. Jeho specialita spočívá v tom, že je řízeno přes paralelní port.

sln Rozhraní SLIP číslo *n*. Rozhraní SLIP jsou přidělena sériovým linkám v takovém pořadí, v jakém jsou sériové linky přiřazovány protokolu SLIP; například první sériová linka nakonfigurovaná pro protokol SLIP bude *sl0* atd. Jádro podporuje až čtyři rozhraní SLIP.

pppn Rozhraní PPP číslo *n*. Stejně jako rozhraní SLIP je i rozhraní PPP přiděleno sériové lince v okamžiku, kdy je sériová linka přepnuta do režimu PPP. V současné době jsou podporována až čtyři rozhraní.

plipn Rozhraní PLIP číslo *n*. Protokol PLIP dopravuje datagramy po paralelních linkách. Jsou podporována až tři rozhraní PLIP. Tato rozhraní jsou přidělována ovladačem PLIP při procesu zavádění systému a jsou mapována na paralelní porty.

Pro ostatní ovladače rozhraní, které budou připsány později, například ISDN nebo AX.25, budou zavedeny nové názvy.

V následujících odstavcích si popíšeme použití výše popsaných ovladačů.

3.4 Instalace Ethernetu

Aktuální síťový kód Linuxu podporuje různé skupiny ethernetových karet. Většinu ovladačů napsal Donaldem Becker (**becker@cesdis.gsfc.nasa.gov**), který je autorem rodiny ovladačů karet založených na čipu National Semiconductor 8390; tato skupina se proslavila pod názvem Becker Series Drivers. Dále existují ovladače pro několik produktů od firmy D-Link, mezi ně patří i kapesní adaptér D-Link, který umožňuje přistupovat k Ethernetu přes paralelní port. Tento ovladač napsal Bjřrn Ekwall (**bj0rn@blox.se**). Ovladač DEPCA vytvořil Davide C. Davies (**davies@wanton.lkg.dec.com**).*

3.4.1 Kabeláž Ethernetu

Pokud instalujete Ethernet poprvé v životě, pak se vám možná bude hodit krátký úvod do kabeláže. Ethernet je velmi vybíravý na správné kabely. Kabel musí být na obou koncích ukončen rezistorem o odporu 50 Ohmů a nesmí mít žádné větvení (například tři kabely spojené do hvězdy). Pokud používáte tenký koaxiální kabel s BNC-konektory tvaru T, musí být tyto konektory zacvaknuty přímo do konektoru karty; neměli byste vřazovat žádný kabelový segment.

Pokud používáte tlusté kabely, musíte svého hostitele připojit přes tzv. transceiver (někdy se také používá označení jednotka pro připojení Ethernetu). Transceiver můžete zapojit do 15pólového portu AUI, který najdete na vaší kartě. Alternativou je použití stíněného kabelu.

3.4.2 Podporované karty

Kompletní seznam podporovaných karet je k dispozici v dokumentu Ethernet HOWTO, jehož autorem je Paul Gotmaker.

Následuje seznam známějších karet podporovaných systémem Linux. Aktuální seznam, který najdete v dokumentu HOWTO, je asi třikrát delší. I když ale najdete v tomto seznamu svou kartu, podívejte se nejprve do dokumentu HOWTO. Někdy jsou tam totiž důležité informace o tom, jak tyto karty úspěšně zprovoznit. To se týká zejména některých ethernetových karet založených na kanálu DMA, které používají stejný DMA-kanál jako implicitně nastavený SCSI-řadič Adaptec 1542. Pokud nepřesunete jednu z karet na jiný DMA-kanál, bude ethernetová karta zapisovat data z paketu na libovolná místa na vašem pevném disku.

3Com EtherLink

Jsou podporovány karty 3c503 a 3c503/16, stejně tak i 3c507 a 3c509. To platí i pro kartu 3c501, která je ale příliš pomalá, než aby se její koupě vyplatila.

* Poznámka korektora: Linux dnes podporuje naprostou většinu síťových karet.

Novell Eagle

NE1000, NE2000 a spousta jejich klonů. Jsou podporovány i karty NE1500 a NE2100.

Western Digital/SMC

Podporovány jsou karty WD8003 a WD8013 (stejné jako SMC Elite a SMC Elite Plus) a také nová karta SMC Elite 16 Ultra.

Hewlett Packard

HP 27252, HP 27247B a HP J2405A.

D-Link

Kapesní adaptér DE-600, DE-100, DE-200 a DE-220-T. Existuje i záloha pro kartu DE-650-T, což je PCMCIA karta.[4]

DEC

DE200 (32K/64K), DE202, DE100 a DEPCA model E.

Allied Telesis

AT1500 a AT1700.

V Linuxu lze společně s jednou z těchto karet použít předkompilované jádro od jednoho z hlavních linuxových distributorů. Tato jádra mají zpravidla vestavěné ovladače pro všechny typy karet. Nicméně z dlouhodobého hlediska je lepší si sestavit své vlastní jádro a zkompilovat ho pouze s těmi ovladači, které budete skutečně potřebovat.

3.4.3 Automatická detekce Ethernetu

Při zavádění se ethernetový kód pokusí nalézt vaši kartu a určit její typ. Karty jsou detekovány na následujících adresách a v následujícím pořadí:

Karta	Sondované adresy
WD/SMC	0x300, 0x280, 0x380, 0x240
SMC 16 Ultra	0x300, 0x280
3c501	0x280

[4] Společně s ostatním materiálem pro přenosné počítače ji lze získat na adrese **tsx-11.mit.edu** v adresáři `packages/laptops`.

Karta	Sondované adresy
3c503	0x300, 0x310, 0x330, 0x350, 0x250, 0x280, 0x2a0, 0x2e0
Nex000	0x300, 0x280, 0x320, 0x340, 0x360
HP	0x300, 0x320, 0x340, 0x280, 0x2c0, 0x200, 0x240
DEPCA	0x300, 0x320, 0x340, 0x360

Pro kód automatické detekce existují dvě omezení. Za prvé, jádro nemusí korektně rozpoznat všechny karty. To platí zejména pro některé levnější klony běžných karet, ale také pro některé karty WD80x3. Druhým problémem je, že jádro nebude automaticky detekovat výskyt více než jedné karty. To je záměr, protože se předpokládá, že budete osobně chtít řídit, které rozhraní bude dané kartě přiřazeno.

Používáte-li více než jednu kartu nebo pokud se automatická detekce karty nezdaří, je třeba jádru explicitně říci základní adresu a název karty.

V Net-3 můžete k tomuto účelu použít dva různé postupy. Jeden spočívá v úpravě nebo doplnění informací do souboru `drivers/net/Space.c` se zdrojovým kódem jádra, který obsahuje veškeré informace o ovladačích. Tento způsob se doporučuje pouze v případě, že jste obeznámeni se síťovým kódem. Mnohem lepší je poskytnout jádru tyto informace při zavádění. Pokud pro zavádění systému použijete příkaz `lilo`, můžete předat parametry jádra za pomoci volby `append` v souboru `lilo.conf`. Chcete-li jádro informovat o ethernetovém zařízení, předejte mu následující parametr:

```
ether=irq,base_addr,param1,param2,name
```

První čtyři parametry jsou číselné, zatímco poslední představuje název zařízení. Všechny numerické hodnoty jsou nepovinné; pokud je vynecháte nebo uvedete nulové hodnoty, pokusí se jádro získat jejich hodnotu detekcí nebo místo nich použije implicitní hodnoty.

První parametr nastavuje IRQ, které bude přiděleno zařízení. Jádro se implicitně pokusí automaticky detekovat IRQ-kanál daného zařízení. Ovladač 3c503 disponuje speciální vlastností, která vybere volný IRQ-kanál ze seznamu 5, 9, 3, 4 a nakonfiguruje kartu tak, aby tento kanál používala.

Parametr **`base_addr`** udává V/V základní adresu karty. Zadáte-li hodnotu nula, pokusí se jádro zjistit adresu z výše uvedeného seznamu.

Zbývající dva parametry mohou různé typy ovladačů používat odlišným způsobem. U karet sdílejících paměť, jako například WD80x3, určují počáteční a koncovou adresu oblasti sdílené paměti. Ostatní karty používají zpravidla parametr **`param1`** k nastavení úrovně zobrazení ladicích informací. Hodnoty od 1 do 7 ukazují zvyšující se úroveň rozsahu výpisů, zatímco

hodnota 8 ji vypíná; 0 udává implicitní volbu. Ovladač 3c503 používá **param2** k výběru vnitřního transceiveru (implicitně) nebo vnějšího transceiveru (hodnota 1). Nulová hodnota způsobí použití BNC-konektoru umístěného na kartě; hodnota 1 vyvolá použití portu AUI.

Pokud máte dvě ethernetové karty, může Linux automaticky detekovat jednu z nich a parametry druhé karty mu pak předáte pomocí příkazu `lilo`. Je třeba se však ujistit, že ovladač náhodně nenašel nejprve druhou kartu. V tom případě by totiž druhou kartu vůbec nenašel. Lze tak učinit pomocí volby **reserve** v příkazu `lilo`, která explicitně řekne jádru, aby nedetekoval V/V-prostor obsazený druhou kartou.

Chcete-li například, aby Linux nainstaloval druhou ethernetovou kartu na adresu **0x300** jako *eth0*, musíte jádru předat následující parametry:

```
reserve=0x300,32 ether=0,0x300,eth1
```

Volba `reserve` zajistí, aby žádný ovladač nepřistupoval při detekci určitého zařízení k V/V-prostoru karty. Parametry jádra lze také použít k přesměrování automatické detekce *eth0*:

```
reserve=0x340,32 ether=0,0x340,eth0
```

Chcete-li úplně vypnout automatickou detekci, použijte argument **base_addr** s hodnotou –1:

```
ether=0,-1,eth0
```


3.5 Ovladač PLIP

Protokol PLIP znamená *IP-protokol po paralelní lince* (*Parallel Line IP*) a představuje levnou alternativu sítě, pokud chcete propojit pouze dva počítače. Používá paralelní port společně se speciálním kabelem a dosahuje rychlosti od 10 kBps do 40 kBps.

Protokol PLIP původně vyvinula firma Crynwr, Inc. Jeho návrh je poměrně prostý: po dlouhou dobu byly u počítačů PC paralelní porty používány pouze jako jednosměrné porty pro tiskárny; to znamená, že při posílání informací z počítače PC do periferního zařízení mohlo být použito pouze osm datových linek. Posílání nebylo možné opačným směrem. Protokol PLIP se s tím vypořádal tak, že u portu používá pro vstup pět stavových linek, což ale zpomaluje přenos dat, protože všechna data jsou přenášena pouze po částech (po polovinách bajtu). Tento pracovní režim se nazývá nulový režim protokolu PLIP. Dnes se zdá, že se již tyto jednosměrné porty vůbec nepoužívají. Proto existuje rozšíření protokolu PLIP zvané režim 1, které používá plné 8bitové rozhraní.

Na rozdíl od předchozích verzí kódu protokolu PLIP se současná verze snaží být kompatibilní s implementacemi protokolu PLIP firmy Crynwr. Totéž platí i pro ovladač protokolu PLIP v `telnetu` NCSA.[5] Ke spojení dvou počítačů za pomoci protokolu PLIP potřebujete speciální kabel, který seženete v některých obchodech pod označením kabel „Null Printer" nebo „Turbo Laplink". Poměrně jednoduše si ale můžete vyrobit kabel vlastní. Příloha A ukazuje, jak na to.

Linuxový ovladač protokolu PLIP je výsledkem práce obrovského množství lidí. Momentálně ho spravuje pan Niibe Yutaka. Pokud je ovladač protokolu PLIP zakompilován do jádra, nastaví následující síťové rozhraní každého z dostupných paralelních portů: rozhraní *plip0* bude odpovídat paralelnímu portu `lp0`, rozhraní *plip1* bude odpovídat paralelnímu portu `lp1` atd. Mapování rozhraní PLIP k paralelním portům je následující:

Rozhraní	V/V port	IRQ
plip0	0x3BC	7
plip1	0x378	7
plip2	0x278	5

Máte-li port pro tiskárnu nakonfigurován odlišným způsobem, je třeba upravit tyto hodnoty v souboru `drivers/net/Space.c`, ve zdrojovém kódu jádra Linuxu. Potom je třeba jádro znovu sestavit.

Nicméně toto mapování neznamená, že nemůžete používat paralelní porty obvyklým způsobem. Ovladač protokolu PLIP přistupuje k paralelním portům pouze v případě, že je k němu nakonfigurováno odpovídající rozhraní.

3.6 Ovladače SLIP a PPP

Protokoly SLIP (IP po sériové lince) a PPP (Protokol Point-to-Point) se hojně využívají k posílání IP-paketů po sériové lince. Připojení SLIP a přístup k internetovým počítačům za pomoci protkolu PPP nabízí množství firem. Tímto způsobem poskytují IP připojení soukromým osobám (jinak by pro ně bylo připojení cenově nedostupné).

Abyste mohli používat ovladače SLIP nebo PPP, nejsou nutné žádné hardwarové úpravy; stačí použít libovolný sériový port. Protože sériové porty nejsou pro komunikaci na bázi TCP/IP typické, bude jim věnována samostatná kapitola.Více informací tak najdete v kapitole 4.

[5] `Telnet` NCSA je populární program pro DOS, který spouští protokol TCP/IP pomocí Ethernetu nebo protokolu PLIP a podporuje standardy `telnet` a FTP.

Kapitola

4

Nastavení sériového hardwaru

Povídá se, že v oblasti sítí stále ještě existují lidé, kteří vlastní pouze jediný počítač PC a kteří nemají dostatek peněz na internetové spojení typu T1. Aby však získali svou „každodenní dávku konferencí a pošty“, spoléhají na spojení pomocí protokolu SLIP, na sítě typu UUCP a na systémy BBS, které využívají veřejné telefonní sítě.

Tato kapitola má pomoci všem lidem, kteří jsou odkázáni pouze na modemy. Do přílišných podrobností se ale v této kapitole pouštět nemůžeme, například jak nakonfigurovat modem pro volání. Všechna tato témata budou zpracována v dokumentu Grega Hankinse Serial HOWTO[1].

4.1 Komunikační software pro modemová spojení

V Linuxu je k dispozici mnoho komunikačních balíků. Spoustu z nich tvoří *terminálové programy*, které umožňují uživateli propojení s jiným počítačem a které se tváří, jakoby uživatel seděl před jednoduchým terminálem. Tradičním unixovým terminálovým programem je `kermit`. Je však značně zastaralý. Dnes jsou dostupné mnohem komfortnější programy, které podporují adresář s telefonními čísly, skriptové jazyky, jež se používají pro volání a připojování ke vzdáleným počítačovým systémům atd. Jeden z nich se nazývá `minicom`. Je podobný některým terminálovým programům, které možná znají dřívější uživatelé operačního systému DOS. Existují také komunikační balíky na bázi X, například `seyon`.

Rovněž je k dispozici množství linuxových balíků pro BBS. Některé z těchto balíků je možné nalézt na FTP-serveru **sunsite.unc.edu** v adresáři `/pub/Linux/system/Network`.

---

1 Greg Hankins je k zastižení na e-mailu **gregh@cc.gatech.edu**.

Kromě terminálových programů existuje software, který při přenosu dat z vašeho nebo do vašeho počítače nevyužívá sériovou linku interaktivním způsobem. Výhodou této metody je výrazně kratší čas potřebný k automatickému stažení několika desítek kilobajtů. Typickým příkladem je on-line čtení pošty z poštovní schránky nebo hledání zajímavých článků v systému BBS. Na druhou stranu však tento způsob vyžaduje mnohem větší diskový prostor, protože načítá i bezcenné informace.

Typickým představitelem tohoto typu softwaru je UUCP. Jedná se o programový balík, který kopíruje data z jednoho hostitele na druhý, spouští programy na vzdáleném hostiteli atd. V soukromých sítích je často používán pro přenos pošty nebo konferencí. V následující kapitole bude popsán balík UUCP od Iana Taylora, který také běží v prostředí Linuxu. Další neinteraktivní komunikační software se používá například v sítích Fidonet. Jsou k dispozici i aplikace přenesené na platformu Linuxu, například `ifmail`.

Internetový protokol pro sériové linky SLIP se nachází někde na rozmezí, dovoluje jak interaktivní, tak i neinteraktivní použití. Mnoho lidí využívá protokol SLIP pro připojení ke školní síti nebo k některému jinému druhu veřejného SLIP-serveru. Mohou tak využívat služby protokolu FTP a jiné. Protokol SLIP lze ale použít také k propojení několika lokálních sítí pevnými nebo částečně pevnými linkami. Toto využití je zajímavé pouze pokud současně použijeme zařízení ISDN.

4.2 Úvod k sériovým zařízením

Zařízení, pomocí nichž umožňuje jádro Unixu přístup k sériovým zařízením, se obvykle nazývají *tty*. Je to zkratka názvu společnosti *Teletype*(TM), která bývala v počátcích unixové éry jedním z hlavních výrobců terminálů. V současnosti se tento termín používá pro jakékoliv znakově založené datové terminály. V průběhu této kapitoly budeme tento termín používat výhradně k označení zařízení jádra.

Linux rozlišuje tři třídy tty: (virtuální) konzoly, pseudoterminály (podobné obousměrnému potrubí, které používají aplikace jako je X11) a sériová zařízení. Posledně zmíněná třída je rovněž považována za tty, poněvadž umožňuje interaktivní spojení po sériové lince; sériová zařízení lze používat s využitím telefonní linky buďto z pevně připojeného terminálu, nebo ze vzdáleného počítače.

Zařízení tty mají spoustu konfiguračních parametrů, které lze nastavovat pomocí systémového volání *ioctl(2)*. Mnoho z nich se vztahuje pouze na sériová zařízení, protože ty vyžadují ke správě různých typů spojení výrazně vyšší flexibilitu.

Mezi nejvýznamnější parametry linek patří rychlost a parita. Existují však i tzv. registry, které řídí konverzi mezi malými a velkými písmeny, převod znaků CR (carriage return) na LF (linefeed) atd. Ovladač tty může také podporovat různé linkové disciplíny, které mohou zcela změnit chování ovladače zařízení. Například linuxový ovladač SLIP je implementován jako speciální linková disciplína.

Pro způsob měření rychlosti linky je příznačná určitá dvojznačnost. Správným termínem je tzv. *bitová rychlost*, která se vztahuje na rychlost přenosu linky a měří se v bitech za sekundu (zkráceně bps). Někdy lidé tento termín označují jako tzv. *přenosovou rychlost*, což není tak úplně správné. Tyto termíny totiž nelze zaměňovat. Přenosová rychlost se vztahuje k fyzickým charakteristikám daného sériového zařízení, konkrétně k taktovacímu kmitočtu, ve kterém jsou přenášeny pulsy. Bitová rychlost označuje spíše aktuální stav existujícího sériového spojení mezi dvěma body, konkrétně průměrný počet bitů přenesený za sekundu. Je důležité vědět, že tyto dvě hodnoty se obvykle liší, protože většina zařízení kóduje do elektrického impulsu více než jeden bit.

4.3 Přístup k sériovým zařízením

Stejně jako ke všem unixovým zařízením, je i k sériovým zařízením přistupováno pomocí speciálních souborů zařízení, které se nachází v adresáři `/dev`. K ovladačům sériových zařízení se vztahují dvě skupiny souborů zařízení. Každému portu odpovídá jeden soubor zařízení z každé skupiny souborů zařízení. Zařízení se bude chovat v závislosti na typu souboru, který se používá pro přístup k tomuto zařízení.

První skupina souborů zařízení se použije kdykoliv, kdy je k portu přistupováno ve směru do počítače; má hlavní číslo 4 a soubory se nazývají `ttyS0`, `ttyS1` atd. Druhá skupina se použije pro vytáčení směrem z počítače; soubory se nazývají `cua0` atd. a jejich hlavní číslo je rovno 5.

Vedlejší čísla jsou pro oba typy stejná. Máte-li modem připojen na jednom z portů *COM1* až *COM 4*, bude vedlejší číslo rovno číslu *COM*-portu plus 63. Pokud se vaše nastavení liší nebo používáte například kartu, která podporuje několik sériových linek, podívejte se, prosím, do dokumentu Serial HOWTO.

Budeme předpokládat, že váš modem je připojen na port *COM2*. Jeho vedlejší číslo tak bude mít hodnotu 65 a hlavní číslo bude rovno 5, protože modem budeme používat pro volání z počítače. Mělo by existovat zařízení `cva1`, které bude mít tyto hodnoty. Vypište si sériová zařízení tty z adresáře `/dev`. Sloupce 5 a 6 by měly ukazovat hlavní a vedlejší čísla v uvedeném pořadí:

```
$ ls -l /dev/cua*
crw-rw-rw-   1 root     root     5,  64 Nov 30 19:31 /dev/cua0
crw-rw-rw-   1 root     root     5,  65 Nov 30 22:08 /dev/cua1
crw-rw-rw-   1 root     root     5,  66 Oct 28 11:56 /dev/cua2
crw-rw-rw-   1 root     root     5,  67 Mar 19  1992 /dev/cua3
```

Jestliže takové zařízení neexistuje, musíte je vytvořit: přihlaste se jako superuživatel a napište:

```
# mknod -m 666 /dev/cua1 c 5 65
# chown root.root /dev/cua1
```

Někteří lidé doporučují vytvořit soubor `/dev/modem`, který by sloužil jako symbolické spojení s vaším modemovým zařízením, takže si příležitostní uživatelé nemusí pamatovat poněkud neintuitivní `cua1`. Nemůžete ale v jednom programu používat soubor `modem` a ve druhém skutečný název souboru zařízení. To proto, že programy používají tzv. *zamykací soubory (lock file)*, které signalizují, že dané zařízení je právě používáno. Podle úmluvy je například název zamykacího souboru pro `cua1` `LCK..cua1`. Použijete-li pro stejný port různé názvy souborů zařízení, pak programy správně nerozliší zamykací soubory a budou současně přistupovat ke stejnému zařízení. Výsledkem bude, že nebude pracovat ani jedna z aplikací.

4.4 Sériový hardware

V současné době podporuje Linux širokou škálu sériových karet, které využívají standard RS-232C, který je momentálně nejběžnějším pro sériovou komunikaci ve světě počítačů PC. K přenosu jednotlivých bitů a synchronizaci využívá několik elektronických obvodů. Další linky lze využít k signalizaci výskytu tzv. nosné (carrier) (kterou používají modemy) a k signalizaci počátku výměny informací, tzv. handshake (podání ruky).

Ačkoliv je hardwarový handshake volitelný, je velice užitečný. Umožňuje oběma stanicím vzájemné předávání informací o stavech - zda je jedna strana připravena na příjem dalších dat nebo zda by měla druhá strana počkat, až strana, která je na příjmu, dokončí zpracování příchozích dat. K tomuto účelu se používají tzv. linky „Clear to Send“ (CTS) a „Ready to Send“ (RTS), které provádějí cosi na způsob hardwarového handshake, konkrétně „RTS/CTS“.

2 Existoval i čip NSC 16 550, ale jeho paměť FIFO snad vůbec nikdy nefungovala.

V počítačích PC je rozhraní RS-232 obvykle řízeno čipem UART, který je odvozen od čipu 16 450 firmy National Semiconductor nebo od jeho novější verze NSC 16 550A[2]. Některé skupiny zařízení (nejvýraznějšími z nich jsou interní modemy vybavené čipovou sadou firmy Rockwell) používají ale úplně odlišné čipy, které byly naprogramovány tak, aby se chovaly jako čip 16 550.

Hlavní odlišností čipů 16 450 a 16 550 je podpora vyrovnávací paměti FIFO o velikosti 16 bajtů, která je implementována v novějším čipu 16 550, zatímco čip 16 450 disponuje pouze vyrovnávací pamětí o velikosti 1 bajt. Tento rys omezuje použití čipu 16 450 pro maximální přenosové rychlosti 9 600 bps, ale čip kompatibilní s 16 550 lze použít i pro rychlosti vyšší. Kromě těchto čipů podporuje Linux i čip 8250, což byl původní čip počítačů PC AT.

Jádro implicitně kontroluje čtyři standardní sériové porty *COM1* až *COM4*. Těmto portům jsou přidělena vedlejší čísla v rozmezí od 64 do 67, viz výše uvedený popis.

Pro přesné nastavení sériových portů slouží příkaz *setserial,* jehož autorem je Ted Ts'o, a s ním i skript `rc.serial`. Tento skript lze vyvolat při zavádění systému z adresáře `/etc/rc`. Ke konfiguraci sériových zařízení jádra používá skript `rc.serial` příkaz `setserial`. Typický skript `rc.serial` vypadá následovně:

```
# /etc/rc.serial - Konfigurační skript sériové linky
#
# Detekce přerušení
/sbin/setserial -W /dev/cua*

# Konfigurace sériových zařízení
/sbin/setserial /dev/cua0 auto_irq skip_test autoconfig
/sbin/setserial /dev/cua1 auto_irq skip_test autoconfig
/sbin/setserial /dev/cua2 auto_irq skip_test autoconfig
/sbin/setserial /dev/cua3 auto_irq skip_test autoconfig

# Zobrazení konfigurace sériových zařízení
/sbin/setserial -bg /dev/cua*
```

Vysvětlení parametrů najdete v dokumentaci příkazu `setserial`.

Pokud nebude detekována vaše sériová karta, nebo pokud příkaz `setserial -bg` zobrazí nesprávná nastavení, je třeba konfiguračnímu skriptu explicitně poskytnout správné hodnoty. Uživatelé interních modemů s čipovou sadou firmy Rockwell potvrzují, že se s tímto problémem již setkali. Pokud bude například čip UART detekován jako čip NSC 16 450, i když je ve skutečnosti kompatibilní s čipem NSC 16 550, je nutné změnit konfigurační příkaz na:

```
/sbin/setserial /dev/cua1 auto_irq skip_test autoconfig uart 16550
```

Podobné volby existují i pro specifikaci konkrétního *COM*-portu, základní adresy a nastavení IRQ. Podívejte se prosím do manuálu příkazu `setserial(8)`.

Pokud váš modem podporuje hardwarový handshake, měli byste se ujistit, že je opravdu povolen. Protože většina komunikačních programů předpokládá, že je hardwarový handshake povolen, nesnaží se ho implicitně zapínat; takže je třeba to provést ručně. Nejlépe je to provést ve skriptu `rc.serial` za pomoci příkazu *stty*:

```
$ stty crtscts < /dev/cua1
```

Ke kontrole aktivity hardwarového handshake použijte příkaz:

```
$ stty -a < /dev/cua1
```

Tento řádek vypíše stav všech registrů příslušného zařízení; znaménko minus předcházející zobrazenému registru tento registr vypíná, například **`-crtscts`**.

Kapitola

5

Konfigurace sítí na bázi TCP/IP

V této kapitole si projdeme kroky, které jsou nezbytné pro konfiguraci sítí na bázi protokolu TCP/IP. Začneme s přidělením IP-adresy, pak si projdeme nastavení rozhraní TCP/IP a představíme si několik užitečných nástrojů pro hledání problémů ve vaší síťové instalaci.

Většinu úkolů probraných v této kapitole budete muset obvykle provést pouze jedenkrát. Pozdější změny v konfiguračních souborech budou nutné jen v případech, kdy budete chtít do sítě přidat nový systém; nebo budete-li chtít systém kompletně překonfigurovat. Nicméně některé z příkazů pro konfiguraci protokolu TCP/IP musí být spouštěny pokaždé při zavádění systému. To se zpravidla provádí jejich voláním ze systémových skriptů v adresáři `/etc/rc`.

Část procedury týkající se sítí je většinou obsažena ve skriptu s názvem `rc.net` nebo `rc.inet`. Někdy se také můžete setkat se dvěma skripty s názvy `rc.inet1` a `rc.inet2`. První inicializuje síťovou část jádra systému, zatímco druhý spouští základní síťové služby a aplikace. V průběhu následujícího výkladu se budeme držet druhého zmíněného konceptu.

Dále si vysvětlíme akce prováděné skriptem `rc.inet1`, vlastní aplikace budou probrány v pozdějších kapitolách. Po přečtení této kapitoly byste měli umět napsat příslušnou posloupnost příkazů, která správně nakonfiguruje síť s protokolem TCP/IP. Dále byste měli nahradit všechny vzorové příkazy v souboru `rc.inet1` svými vlastními příkazy a ujistit se, že dochází ke spouštění tohoto skriptu při zavádění, a restartovat počítač. Síťové *rc*-skripty dodávané společně s oblíbenou verzí Linuxu vám mohou posloužit jako dobré příklady.

5.1 Nastavení souborového systému proc

Některé z konfiguračních nástrojů balíku Net verze 2 spoléhají při komunikaci s jádrem na souborový systém `proc`. Jedná se o rozhraní, které umožňuje přístup k aktuálním informacím jádra za pomoci mechanismu, jenž je podobný souborovému systému. Když je systém `proc` připojen, můžete stejně jako u ostatních souborových systémů vypisovat jeho soubory nebo zobrazovat jejich obsah. Typickými položkami jsou soubory *loadavg*, které obsahují průměrné zatížení systému, nebo soubor `meminfo`, který zobrazuje aktuální obsazení fyzické paměti a využití odkládacích souborů.

K tomuto účelu přidává síťový kód adresář `net`. Obsahuje množství souborů s informacemi jako například tabulky ARP jádra systému, stav spojení na bázi TCP a směrovací tabulky. Většina síťových administrativních nástrojů čerpá informace z těchto souborů.

Souborový systém `proc` (někdy se používá označení *procfs*) je obvykle připojen při zavádění systému jako adresář `/proc`. Nejlépe toho dosáhnete přidáním následujícího řádku do souboru `/etc/fstab`:

```
# připojovací bod procfs
none /proc proc defaults
```

Potom ze svého skriptu `/etc/rc` spusťte příkaz „**`mount / proc`**“.

V současné době je souborový systém *procfs* implicitně začleněn do většiny jader operačního systému. Není-li souborový systém *procfs* ve vašem jádru, objeví se například zpráva „**mount: fs type procfs not supported by kernel**“. V tom případě budete muset překompilovat jádro systému a na dotaz, zda si přejete nainstalovat podporu souborového systému *procfs*, odpovědět „yes“.

5.2 Instalace binárních souborů

Používáte-li jednu z předkompilovaných distribucí Linuxu, bude velmi pravděpodobně obsahovat hlavní síťové aplikace a utility, a také kompletní sadu vzorových souborů. Jediným případem, kdy budete chtít získat a nainstalovat nové utility, je instalace nové verze jádra operačního systému. Protože to občas vede ke změnám v síťové vrstvě jádra systému, budete muset aktualizovat i základní konfigurační nástroje. To znamená přinejmenším rekompilaci binárních souborů, ale někdy budete potřebovat i nejnovější sady binárních souborů. Ty jsou obvykle distribuovány společně s jádrem a zabaleny v archívu `net-XXX.tar.gz`, kde *XXX* označuje číslo verze. Verze odpovídající Linuxu 1.0 je 0.32b, avšak poslední jádro operačního systému (v době vzniku této publikace to byla verze 1.1.12) vyžaduje verzi 0.32d.

Pokud si chcete sami zkompilovat a nainstalovat standardní síťové aplikace na bázi protokolu TCP/IP, získáte jejich zdrojový kód na většině linuxových serverů. Jsou to více či méně opravené verze programů z balíku Net-BSD nebo z ostatních zdrojů. Ostatní aplikace, jako například `Xmosaic`, `xarchie` nebo Gopher a klienti IRC, je nutné získat samostatně.. Budete-li dodržovat příslušné instrukce, zkompiluje se většina z těchto aplikací takřkajíc po vytažení z krabice.

Oficiální systém pro balík Net-3 najdete na adrese **sunacm.swan.ac.uk**, jehož zrcadlem je **sunsite.unc.edu** a příslušný adresář se jmenuje `system/Network/sunacm`. Poslední aktualizace balíku Net-2e a jeho binární soubory jsou k dispozici na adrese **ftp.aris.com**. Balík BSD od Matthiase Urlichse – odvozený ze síťového kódu - můžete získát na adrese **ftp.ira.uka.de** v adresáři `/pub/system/linux/netbsd`.

5.3 Další příklad

Ve zbytku této knihy si dovolím uvést nový příklad, jenž je oproti případu Groucho Marx University méně složitý, ale může se více podobat problémům, s nimiž se ve skutečnosti setkáte. Vezměme si společnost Virtual Brewery, což je malá společnost, která, jak název napovídá, vaří virtuální pivo. Aby mohli virtuální pivovarníci efektivněji spravovat svůj obchod, chtějí mít své počítače propojeny do sítě. Všechny počítače jsou typu PC se šikovným operačním systémem Linux verze 1.0.

Na stejném podlaží na druhém konci budovy existuje obchod Virtual Winery, který s pivovarem velmi blízce spolupracuje. Mají svůj vlastní Ethernet. Tyto dvě společnosti chtějí mít zcela přirozeně z provozních důvodů své sítě propojeny. Nejprve chtějí nastavit bránu, jenž bude doručovat datagramy mezi těmito dvěma podsítěmi. Dále budou chtít být v kontaktu s okolním světem spojením typu UUCP, pomocí nějž budou přijímat konference a poštu. Z dlouhodobého hlediska budou chtít mít nastavené spojení typu SLIP, aby se mohli příležitostně připojit na Internet.

5.4 Nastavení názvu hostitele

Většina, ne-li všechny aplikace, spoléhají na to, že je název místního hostitele nastaven na nějakou rozumnou hodnotu. To se obvykle provede při zavádění pomocí příkazu *hostname*. Chcete-li nastavit název hostitele například na **name**, zadejte:

```
# hostname name
```

U tohoto příkazu je běžné používat nekvalifikované názvy hostitele bez jakéhokoliv názvu domény. Například hostitelé ve Virtual Brewery se mohou nazývat **vale.vbrew.com**, **vlager.vbrew.com** atd. Toto jsou oficiální, plně kvalifikované názvy domén. Názvy jejich místních hostitelů bude tvořit pouze první část z názvu, například **vale**. Protože je však název lokálního hostitele často používán pro vyhledání IP-adresy hostitele, musíte se ujistit, že knihovna resolveru je schopná vyhledat IP-adresu hostitele. To obvykle znamená, že musíte název hostitele specifikovat v souboru `/etc/hosts` (viz níže). Někteří lidé doporučují používat příkaz `domainname`. Tento příkaz se používá pro informování jádra operačního systému o názvu domény zbývající části FQDN. U tohoto způsobu byste měli pro opětovné získání FQDN zkombinovat výstupy příkazů `hostname` a `domainname`. Tento způsob je však jen z poloviny správný. Příkaz `domainname` se všeobecně používá k nastavení NIS-domény hostitele, která může být zcela jiná než DNS-doména, ke které patří váš hostitel. Doména NIS je probírána v kapitole 10.

5.5 Přidělení IP-adres

Chcete-li nakonfigurovat síťový software na svém hostiteli, který není určen pro práci v síti (aby na něm mohl být například spouštěn software síťových konferencí INN), můžete tuto stať s klidným svědomím přeskočit, protože k tomu budete potřebovat pouze IP-adresu svého zpětnovazebného rozhraní, a ta je vždy **127.0.0.1**.

Mírně složitější je tento problém v reálných sítích typu Ethernet. Chcete-li připojit svého hostitele k existující síti, musíte požádat jejího správce, aby vám v této síti přidělil IP-adresu. Nastavujete-li celou síť sami, musíte si také sami přidělit IP-adresy, což probereme později.

Hostitelé, kteří patří do místní sítě, by měli obvykle sdílet adresy ze stejné logické sítě IP. Z toho důvodu musíte přidělit síťovou IP-adresu. Máte-li několik fyzických sítí, musíte jim buď přidělit odlišná síťová čísla, anebo musíte použít podsítě, čímž rozdělíte svůj rozsah IP-adres do několika podsítí.

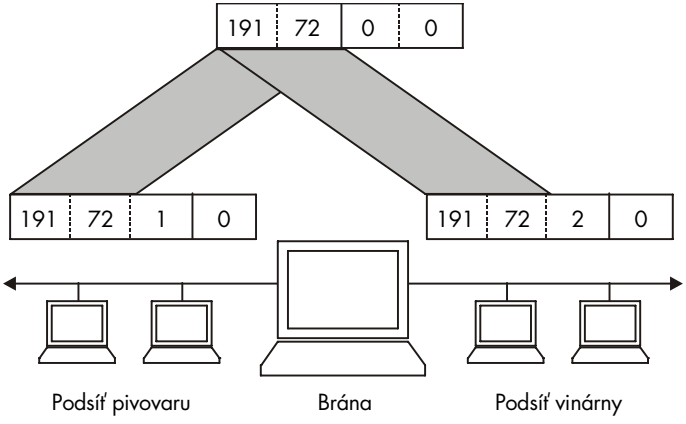
Není-li vaše síť připojena k Internetu, můžete si vybrat libovolnou (platnou) adresu sítě. Pouze se musíte ujistit, že jste vybrali jednu ze tříd A, B nebo C, protože jinak by pravděpodobně vše nefungovalo zcela správně. Plánujete-li v blízké době připojení k Internetu, měli byste si už *nyní* obstarat oficiální IP-adresu. Nejlépe je požádat o pomoc svého poskytovatele síťových služeb. Pokud si chcete obstarat číslo sítě jenom proto, že se někdy v budoucnosti budete chtít připojit na Internet, vyžádejte si na adrese **hostmaster@internic.net** formulář Network Address Application Form.

Budete-li spravovat několik ethernetových sítí (nebo jiných sítí, jakmile pro ně budou dostupné ovladače), musíte rozdělit vaši síť na podsítě. Všimněte si, že vytváření podsítí je požadováno pouze v případě, že vlastníte více než jednu *vysílací síť* (*broadcast network*);

nepočítaje v to spojení pomocí protokolu PPP. Máte-li například jeden Ethernet a jedno nebo více spojení s vnějším světem pomocí protokolu SLIP, nepotřebujete ve své síti vytvářet podsítě. Důvod bude objasněn v kapitole 7.

Síťový správce pivovaru požádal například centrum NIC o přidělení čísla sítě třídy B, obdržel adresu **191.72.0.0**. Aby si správce přizpůsobil své dva Ethernety k obrazu svému, rozhodl se použít osm bitů z části hostitele jako doplňující bity podsítě. Tato úprava poskytla části hostitele dalších osm bitů, což dovoluje až 254 hostitelů v každé podsíti. Dále správce přidělil pivovaru číslo podsítě 1 a vinárně číslo podsítě 2. Tedy jejich příslušné síťové adresy jsou **191.72.1.0** a **191.72.2.0**. Maska podsítě je **255.255.255.0**.

Bráně **vlager**, což je brána mezi dvěma sítěmi, bylo u obou podsítí přiděleno číslo hostitele 1, takže její IP-adresy jsou **191.72.1.1**, resp. **191.72.2.1**. Obrázek 5.1 ukazuje dvě podsítě a bránu.



Obrázek 5.1
Dvě podsítě – Virtual Brewery a Virtual Winery

Všimněte si, že v tomto případě používáme z ilustračních důvodu síť třídy B; síť třídy C by byla mnohem realističtější. U nového síťového kódu není vytváření podsítí limitováno na celé bajty, takže i třída C může být rozdělena do několika podsítí. Dva bity z části hostitele byste mohli například použít na síťovou masku, čímž byste získali čtyři možné podsítě s až 64 hostiteli na každé z nich.[1]

[1] Poslední číslo je v každé podsíti rezervováno pro tzv. vysílací adresu, takže ve skutečnosti může v každé podsíti existovat jen 63 hostitelů.

5.6 Sestavení souborů hosts a networks

Po vytvoření podsítí byste měli za pomoci souboru `/etc/hosts` připravit několik jednoduchých typů rozlišení názvu hostitele. Pokud nehodláte k rozlišení adres používat DNS nebo NIS, musíte do souboru `hosts` vložit všechny hostitele.

Dokonce i v případě, kdy chcete používat DNS nebo NIS při normálních operacích, budete chtít mít v souboru `/etc/hosts` ze všech názvů hostitelů alespoň nějaké skupiny hostitelů. A nějaký druh rozlišení názvů hostitelů budete požadovat i tehdy, nepoběží-li žádné síťové rozhraní (například při zavádění). Tento způsob není vhodný jen z kvůli většímu pohodlí, ale umožní vám také používat ve skriptech `rc.inet` symbolické názvy hostitelů. Takže při změně IP-adres vám postačí pouze zkopírovat aktualizovaný soubor `hosts` na všechny počítače a nemusíte se zatěžovat editací velkého počtu souborů*.rc* Do souboru `hosts` obvykle přidáte všechny místní názvy hostitelů a jejich adresy a jsou-li používány NIS-servery[2] nebo nějaké brány, přidáte i je.

V průběhu úvodního testování byste se měli také ujistit, že váš resolver používá pouze informace ze souboru `hosts`. Software DNS nebo NIS může mít u sebe ukázkové soubory, které, pokud je použijete, mohou produkovat zvláštní výsledky. Chcete-li, aby všechny aplikace při vyhledání IP-adresy hostitele používaly výhradně soubor `/etc/hosts`, musíte upravit soubor `/etc/host.conf`. Všechny řádky, které začínají klíčovým slovem *order,* označte jako komentáře. Provedete to tak, že na první pozici příslušného řádku vložíte znak (#). Dále vložte následující řádek:

```
order hosts
```

Konfigurace knihovny resolveru bude detailně rozebrána v kapitole 6.

Soubor `hosts` obsahuje na každém řádku jednu položku, která se skládá z IP-adresy, názvu hostitele a z nepovinného seznamu přezdívek názvu hostitele. Jednotlivá pole jsou vzájemně oddělena mezerami nebo tabulátory a pole s adresou musí začínat ve sloupci jedna. Cokoliv, co následuje za znakem #, je považováno za komentář a je ignorováno.

Názvy hostitelů mohou být buď plně kvalifikované, nebo relativní vzhledem k místní doméně. U serveru **vale** zadáte obvykle plně kvalifikovaný název **vale.vbrew.com** a do souboru *hosts* napíšete pouze název **vale**. Tím bude definován jak oficiální název, tak i kratší místní název.

[2] Adresu některých serverů NIS budete potřebovat pouze v případě, že budete používat NYS od Petera Erikssona. Ostatní implementace NIS najdou své servery při spuštění pomocí příkazu `ypbind`.

Následující příklad ilustruje, jak by mohl vypadat soubor *hosts* ve společnosti Virtual Brewery. Jsou v něm obsaženy dva speciální názvy, **vlager-if1** a **vlager-if2**, které poskytují adresy pro obě rozhraní používaná bránou **vlager**.

```
#
# Soubor hosts pro společnosti Virtual Brewery/Virtual Winery
#
# IP          lokální jméno       plně kvalifikované doménové jméno
#
127.0.0.1     localhost
#
191.72.1.1    vlager              vlager.vbrew.com
191.72.1.1    vlager-if1
191.72.1.2    vstout              vstout.vbrew.com
191.72.1.3    vale                vale.vbrew.com
#
191.72.2.1    vlager-if2
191.72.2.2    vbeaujolais         vbeaujolais.vbrew.com
191.72.2.3    vbardolino          vbardolino.vbrew.com
191.72.2.4    vchianti            vchianti.vbrew.com
#
```

Stejně jako u IP-adres hostitelů budete také někdy chtít použít symbolický název pro číslo sítě. Proto má soubor `hosts` doprovodný soubor nazvaný `/etc/networks`, který mapuje názvy sítí na čísla sítí a obráceně. Ve společnosti Virtual Brewery můžeme nainstalovat následující soubor `networks`:[3]

```
# soubor /etc/networks pro společnost Virtual Brewery
brew-net       191.72.1.0
wine-net       191.72.2.0
```

3 Všimněte si, že názvy v souboru `networks` nesmí kolidovat s názvy hostitelů v souboru `hosts`, jinak by mohly některé programy produkovat nesprávné výsledky.

5.7 Konfigurace rozhraní pro protokol IP

Po nastavení hardwaru, které jsme probírali v předchozí kapitole, musíte o těchto zařízeních říci síťovému softwaru jádra. K nastavení síťových rozhraní a inicializaci směrovací tabulky se používá několik příkazů. Tyto úkoly se obvykle provádějí při každém startu počítače ze skriptu `rc.inet1`. Základní konfigurační nástroje se nazývají `ifconfig` (kde „if“ znamená rozhraní - interface) a `route`.

Příkaz `ifconfig` se používá pro zpřístupnění rozhraní síťové vrstvě jádra. Tento proces v sobě zahrnuje přidělení IP-adresy, některých parametrů a aktivaci rozhraní, někdy označovanou jako tzv. „uchopení“. Výraz „aktivní“ znamená, že jádro vysílá a přijímá datagramy pomocí rozhraní IP. Nejjednodušší způsob aktivace rozhraní je následující:

```
ifconfig interface ip-address
```

Výše uvedený příkaz přidělí IP-adresu `ip-address` rozhraní `interface` a aktivuje ho. Všechny ostatní parametry jsou nastaveny na své implicitní hodnoty. Například implicitní maska podsítě je odvozena z IP-adresy podle třídy sítě, například adresa **255.255.0.0** odpovídá adrese třídy B. Příkaz `ifconfig` je detailně popsán na konci této kapitoly.

Příkaz `route` umožňuje přidávat nebo odstraňovat směrování ze směrovací tabulky jádra systému. Může být vyvolán následujícím způsobem

```
route [add|del] target
```

kde argumenty `add` a `del` určují, zda se bude směrování do cíle `target` přidávat nebo se z něj bude odstraňovat.

5.7.1 Zpětnovazebné rozhraní

Jedním z prvních aktivovaných rozhraní je zpětnovazebné rozhraní:

```
# ifconfig lo 127.0.0.1
```

Občas zjistíte, že se místo IP-adresy používá fiktivní název hostitele **localhost**. Příkaz `ifconfig` vyhledá příslušný název v souboru `hosts`, kde by měl být deklarován pro adresu **127.0.0.1**:

```
# Příklad záznamu pro localhost v /etc/hosts
localhost       127.0.0.1
```

Chcete-li si prohlédnout konfiguraci rozhraní, spusťte příkaz `ifconfig` a jako argument zadejte název tohoto rozhraní:

```
$ ifconfig lo
lo       Link encap Local Loopback
         inet addr 127.0.0.1 Bcast [NONE SET] Mask 255.0.0.0
         UP BROADCAST LOOPBACK RUNNING MTU 2000 Metric 1
         RX packets 0 errors 0 dropped 0 overrun 0
         TX packets 0 errors 0 dropped 0 overrun 0
```

Jak vidno, zpětnovazebnému rozhraní byla přidělena síťová maska **255.0.0.0**, protože adresa **127.0.0.1** je adresou třídy A. Dále si můžete všimnout, že rozhraní nemá nastavenu vysílací adresu, protože ta nemá u zpětnovazebného rozhraní žádný význam. Pokud však budete na vašem hostiteli provozovat démona `rwhod`, budete muset nastavit relační adresu zpětnovazebného zařízení, protože jinak by démon `rwhod` nefungoval správně. Nastavení broadcast adresy je vysvětleno dále ve stati „Vše o příkazu `ifconfig`“.

Nyní si můžete začít se svou miniaturní sítí experimentovat. Zatím však ve směrovací tabulce stále chybí položka, která řekne protokolu IP, že může toto rozhraní používat pro směrování k cílové adrese **127.0.0.1**. Tu přidáte následujícím příkazem:

```
# route add 127.0.0.1
```

Opět můžete použít namísto IP-adresy název hostitele **localhost**.

Dále byste měli zkontrolovat, že vše správně funguje, například pomocí použití příkazu `ping`. Příkaz `ping` je síťovým ekvivalentem sonaru[4] a je používán k ověření dostupnosti příslušné adresy a na měření prodlev, které se vyskytují při posílání datagramu tam a zpět. Čas požadovaný na tuto operaci je často uváděn pod označením round trip.

```
# ping localhost
PING localhost (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=32 time=1 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=32 time=1 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=32 time=1 ms
^C

--- localhost ping statistics ---
3 packets transmitter, 3 packets received, 0% packet loss
round-trip min/avg/max = 0/0/1 ms
```

4 Pamatuje si někdo na „Echoes“ od skupiny Pink Floyd?

Spustíte-li příkaz `ping` výše uvedeným způsobem, bude se tento příkaz snažit posílat pakety tak dlouho, dokud ho uživatel nepřeruší. Symbol `^C` označuje místo, kde byla stisknuta kombinace kláves Ctrl C.

Výše uvedený příklad ukazuje, že pakety pro adresu **127.0.0.1** jsou doručovány správně a odpovědi se vracejí příkazu `ping` zpět téměř okamžitě. To naznačuje, že jste při svém prvním nastavení síťového rozhraní uspěli.

Pokud se výstup příkazu `ping` nepodobá výše uvedenému výpisu, pak je tu problém. Zkontrolujte každou chybu, která naznačuje, že některý ze souborů nebyl korektně nainstalován. Zkontrolujte, zda jsou binární soubory příkazů `ifconfig` a `route` kompatibilní s vámi používanou verzí jádra operačního systému a hlavně se podívejte, zda bylo jádro zkompilováno s povolením síťových služeb (to poznáte podle přítomnosti adresáře `/proc/net`). Pokud obdržíte chybovou zprávu „Network unreachable“, pak bude zřejmě chyba v příkazu `route`. Ujistěte se, že používáte stejnou adresu, kterou jste předali příkazu `ifconfig`.

Výše popsané kroky stačí k tomu, abyste mohli na samostatném hostiteli používat síťové aplikace. Jakmile přidáte do souboru `rc.inet1` výše uvedené řádky a ujistíte se, že jsou oba soubory `rc.inet` spouštěny z adresáře `/etc/rc`, můžete počítač restartovat a vyzkoušet různé aplikace. Například příkaz „`telnet localhost`“ by měl s vaším hostitelem navázat spojení typu telnet, které vám nabídne přihlašovací výzvu.

Zpětnovazebné rozhraní je ale užitečné nejen jako příklad vhodný pro knihy o sítích nebo jako testovací prostředek při vývoji. Ve skutečnosti ho používají některé aplikace v průběhu normálních operací.[5] Proto ho musíte nakonfigurovat vždy, bez ohledu na to, zda váš počítač je či není připojen k síti.

5.7.2 Ethernetová rozhraní

Konfigurace ethernetového rozhraní má s nastavením zpětnovazebného rozhraní mnoho společného, pouze ve spojitosti s podsítěmi vyžaduje několik dalších parametrů.

Ve společnosti Virtual Brewery jsme rozdělili síť IP, která byla původně sítí třídy B, na podsítě, které jsou třídy C. Aby se v této změně vaše rozhraní vyznalo, měl by příkaz `ifconfig` vypadat následovně:

```
# ifconfig eth0 vstout netmask 255.255.255.0
```

5 Například všechny aplikace založené na RPC používají při startu zpětnovazebné rozhraní společně s démonem `portmapper` na registraci sebe samých.

Tento příkaz přiřadí rozhraní *eth0* IP-adresu **vstout** (adresa **191.72.1.2**). Pokud bychom vynechali síťovou masku, pak by si ji příkaz `ifconfig` odvodil z třídy sítě IP, ze které vyplyne síťová maska **255.255.0.0**. Rychlé ověření vypíše následující text:

```
# ifconfig eth0
eth 0  Link encap 10 Mps Ethernet Hwaddr 00:00:C0:90:B3:42
       inet addr 191.72.1.2 Bcast 191.72.1.255 Mask 255.255.255.0
       UP BROADCAST RUNNING MTU 1500 Metric 1
       RX packets 0 errors 0 dropped 0 overrun 0
       TX packets 0 errors 0 dropped 0 overrun 0
```

Můžete si všimnout, že příkaz `ifconfig` automaticky nastavil vysílací adresu (výše uvedené pole `Bcast`) na obvyklou hodnotu, což je číslo hostitelovy sítě se všemi nastavenými bity v části hostitele. Také velikost přenosové jednotky (maximální velikost ethernetového paketu, kterou bude generovat jádro pro toto rozhraní) byla nastavena na maximální hodnotu 1500 bajtů. Všechny tyto hodnoty mohou být změněny pomocí speciálních voleb, jež budou popsány dále.

Podobně jako tomu bylo u případu zpětnovazebného rozhraní, musíte nyní nainstalovat směrovací data, která budou jádro informovat o síti, jež je dosažitelná pomocí rozhraní *eth0*. Pro firmu Virtual Brewery byste měli vyvolat příkaz `route` následujícím způsobem:

```
# route add -net 191.72.1.0
```

Na první pohled to vypadá trochu magicky, protože není zcela zřejmé, jak příkaz `route` zjistí, přes které rozhraní má směrovat. Náš trik je ale celkem jednoduchý: jádro operačního systému si ověří všechna rozhraní, která byla v minulosti nakonfigurována a porovná cílovou adresu (v tomto případě **191.72.1.0**) se síťovou částí adresy rozhraní (to znamená, že bude po bitech porovnávat hodnotu získanou z adresy rozhraní a síťové masky). Jediné vyhovující rozhraní bude *eth0*.

K čemu tedy slouží volba `-net`? Používá se z toho důvodu, že příkaz `route` umí obsluhovat jak směrování do sítí, tak i směrování k jednotlivým hostitelům (jak jsme si ukázali u příkladu s názvem **localhost**). Je-li mu předána adresa v tečkové notaci, pokusí se příkaz `route` z prohlédnutých bitů hostitelské části odhadnout, zda se jedná o síť nebo o název hostitele. Je-li hostitelská část adresy rovna nule, bude příkaz `route` předpokládat, že se jedná o síť, v opačném případě ji bude považovat za adresu hostitele. Takto si bude příkaz `route` myslet, že je adresa **191.72.1.0** adresou hostitele a nikoliv číslem sítě. Příkaz `route` nemůže tušit, že používáme podsítě. Tuto informaci mu musíme sdělit explicitně pomocí argumentu `-net`.

Samozřejmě, že vypisování výše uvedeného příkazu `route` je únavné a člověk při něm může udělat chyby. Mnohem pohodlnější je použít názvy sítí, které jsme již nadefinovali v souboru `/etc/networks`. Tento způsob výrazně zlepší čitelnost příkazu `route`; dokonce můžeme vynechat i argument `-net`, protože příkaz `route` již bude vědět, že adresa **191.72.1.0** označuje síť.

```
# route add brew-net
```

Po skončení základních konfiguračních kroků se budete chtít ujistit, že vaše ethernetové rozhraní opravdu funguje správně. Z Ethernetu si vyberte hostitele, například **vlager,** a napište:

```
# ping vlager
PING vlager: 64 byte packets
64 bytes from 191.72.1.1: icmp_seq=0. time=11. ms
64 bytes from 191.72.1.1: icmp_seq=1. time=7. ms
64 bytes from 191.72.1.1: icmp_seq=2. time=12. ms
64 bytes from 191.72.1.1: icmp_seq=3. time=3. ms
^C

----vstout.vbrew.com PING Statistics
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms) min/avg/max = 3/8/12
```

Pokud neuvidíte výstup podobný výše uvedenému, bude zřejmě něco v nepořádku. Dojde-li k neobvyklé ztrátě paketů, bude zřejmě chyba na straně hardwaru, například špatné nebo chybějící terminátory apod. Pokud nepřijmete vůbec žádné pakety, měli byste si zkontrolovat konfiguraci rozhraní pomocí příkazu `netstat`. Statistika paketů zobrazená příkazem `ifconfig` by vám měla říci, zda vůbec byly nějaké pakety poslány rozhraní. Máte-li také přístup ke vzdálenému hostiteli, měli byste zkontrolovat statistiku rozhraní i na tomto počítači. Takto můžete přesně určit, kde se pakety ztratily. Kromě toho byste si měli pomocí příkazu `route` zobrazit směrovací informace a zkontrolovat, zda mají oba hostitelé správnou položku směrování. V případě, že je příkaz `route` vyvolán bez jakýchkoliv dalších argumentů, vytiskne kompletní směrovací tabulku jádra (parametr `-n` pouze zobrazí adresy v tečkové notaci, bez uvedení parametru zobrazí příkaz `route` název hostitele):

```
# route -n
Kernel routing table
Destination  Gateway  Genmask         Flags Metric Ref  Use  Iface
127.0.0.1    *        255.255.255.255 UH    1      0    112  lo
191.72.1.0   *        255.255.255.0   U     1      0    10   eth0
```

Podrobný popis těchto polí najdete dále ve stati Kontrola pomocí příkazu `netstat`. Sloupec `Flag` obsahuje seznam všech symbolů příslušejících každému rozhraní. Aktivní rozhraní mají vždy nastaven symbol `U`, symbol `H` naznačuje, že cílová adresa patří hostiteli. Je-li symbol `H` zobrazen u směrování, míníme tím směrování sítě, pak je nutné společně s příkazem `route` specifikovat i volbu `-net`. To, zda se zadané směrování vůbec používá, zjistíte na základě pole `Use`, které najdete ve druhém sloupci zprava. Jeho hodnota by se mezi dvěma voláními příkazu `ping` měla zvyšovat.

5.7.3 Směrování pomocí brány

V předešlé stati jsme probrali pouze nastavení hostitele v jediné síti Ethernet. Poměrně často se však setkáváme se sítěmi, které jsou vzájemně propojeny pomocí brán. Tyto brány mohou spojovat dva nebo více Ethernetů, ale mohou také poskytovat spojení s okolním světem, a samozřejmě také s Internetem. Abyste mohli využít služeb brán, musíte síťové hladině poskytnout doplňující směrovací informace.

Například Ethernety ve společnostech Virtual Brewery a Virtual Winery jsou propojeny pomocí takovéto brány, tuto funkci konkrétně zastává hostitel **vlager**. Předpokládáme, že hostitel **vlager** již byl nakonfigurován, tudíž nám zbývá přidat pouze další položku do směrovací tabulky hostitele **vstout**, která řekne jádru operačního systému, že všichni hostitelé sítě virtuální vinárny jsou dosažitelní přes bránu **vlager**. Patřičným zaklínadlem příkazu `route` je klíčové slovo `gw`, které sdělí příkazu `route`, že následující argument označuje bránu.

```
# route add wine-net gw vlager
```

Samozřejmě, že každý hostitel v síti vinárny, se kterým chcete hovořit, musí mít odpovídající směrovací položku pro síť pivovaru. V opačném případě byste mohli posílat data pouze z hostitele **vstout** na hostitele **vbardolino**, avšak veškeré odpovědi vrácené hostitelem **vbardolino** by putovaly do koše.

Tento příklad popisuje pouze bránu, která přenáší pakety mezi dvěma izolovanými Ethernety. Nyní budeme předpokládat, že brána **vlager** je také připojena k Internetu (například pomocí doplňujícího spojení s využitím protokolu SLIP). V tom případě budeme po bráně **vlager** požadovat, aby se starala o datagramy putující do *libovolné* cílové sítě, která není identická se sítí pivovaru. To lze provést tak, že označíme bránu **vlager** jako implicitní bránu pro hostitele **vstout**:

```
# route add default gw vlager
```

Název sítě **default** je zkratkou adresy **0.0.0.0**, která označuje implicitní směrování. Tento název nemusíte přidávat do souboru `/etc/networks`, protože je zabudován v příkazu `route`.

Když po aplikaci příkazu `ping` na hostitele, který se nachází za jednou nebo více branami, dojde k velké ztrátě paketů, může to ukazovat na příliš přeplněnou síť. Ztráta paketů není ani tak způsobena odlišnostmi zařízení, jako spíše dočasným špičkovým zatížením předávajících hostitelů, což způsobuje z jejich strany prodlevy případně ztrátu příchozích datagramů.

5.7.4 Konfigurace brány

Konfigurace počítače pro posílání paketů mezi dvěma Ethernety je poměrně přehledná. Budeme předpokládat, že jsme zpět u brány **vlager**, která je vybavena dvěma ethernetovými kartami, z nichž každá je spojena s jednou ze dvou sítí. Pak stačí odděleně nakonfigurovat obě rozhraní, přidělit jim jejich vlastní IP-adresy a to je vše.

Je rozumné přidat informace o obou rozhraních do souboru *hosts*, protože takto získáme pro tato dvě rozhraní šikovné názvy:

```
191.72.1.1      vlager      vlager.vbrew.com
191.72.1.1      vlager-if1
191.72.2.1      vlager-if2
```

Sekvence příkazů `pro nastavení obou rozhraní je následující:`

```
# ifconfig eth0 vlager-if1
# ifconfig eth1 vlager-if2
# route add brew-net
# route add wine-net
```

5.7.5 Rozhraní PLIP

Pokud ke spojení dvou počítačů používáte spojení pomocí protokolu PLIP, budou jednotlivá nastavení mírně odlišná od nastavení použitých při konfiguraci Ethernetu. Spojení pomocí protokolu PLIP se také někdy nazývá spojením typu *point-to-point*, protože se na rozdíl od většiny sítí týká pouze dvou hostitelů („bodů“).

Jako příklad budeme uvažovat počítač v provedení laptop, který mají někteří zaměstnanci společnosti Virtual Brewery. Ten je spojen s bránou **vlager** prostřednictvím protokolu PLIP. Vlastní laptop se nazývá **vlite** a má pouze jediný paralelní port. Při zavádění bude tento port registrován jako rozhraní *plip1*. Abyste spojení aktivovali, musíte nakonfigurovat rozhraní *plip1*[6] za pomoci následujících příkazů:

```
# ifconfig plip1 vlite pointopoint vlager
# route add default gw vlager
```

6 Všimněte si, že pointopoint není překlepem. Skutečně se to takto hláskuje.

První příkaz nastavuje rozhraní a sděluje jádru operačního systému, že se jedná o spojení typu point-to-point, u něhož má vzdálená strana přidělenu adresu **vlager**. Druhý řádek nainstaluje implicitní směrování s využitím hostitele **vlager** jako brány. Na straně brány **vlager** je nutná podobná konfigurace, která spojení zaktivuje (volání příkazu `route` není zapotřebí):

```
ifconfig plip1 vlageg pointopoint vlite
```

Zajímavé je, že rozhraní *plip1* brány **vlager** nemusí mít zvláštní IP-adresu, ale může mu být také přidělena adresa **191.72.1.1**.[7]

Nyní máme vyřešeno směrování z počítače laptop do sítě pivovaru; ale stále ještě nám chybí směrovací cesta z libovolného hostitele pivovaru na počítač **vlite**. Jeden z poměrně nešikovných způsobů spočívá v přidání konkrétního směrování do každé směrovací tabulky hostitele, v níž přiřadíte hostiteli **vlager** funkci brány k hostiteli **vlite**.

```
# route add vlite gw vlager
```

Stojíte-li již tváří v tvář problému s dočasným směrováním, je mnohem výhodnější použít dynamické směrování. Jeden z možných způsobů spočívá v použití směrovacího démona `gated`, který musí být nainstalován na každém hostiteli v síti, aby si mohli dynamicky předávat směrovací informace. Daleko nejjednodušší způsob však počítá s využitím *proxy* ARP. Při nainstalovaném proxy ARP bude brána **vlager** odpovídat na libovolné dotazy ohledně hostitele **vlite** zasláním své vlastní ethernetové adresy. Výsledkem této funkce bude přivolání všech paketů určených pro hostitele **vlite** na bránu **vlager**, která je potom pošle na laptop. K proxy ARP se vrátíme ve stati Kontrola tabulek ARP dále.

Budoucí verze balíku Net-3 bude obsahovat nástroj zvaný `plipconfig`, který bude umožňovat nastavení IRQ používaného paralelního portu. Později by mohl být nahrazen obecnějším příkazem `ifconfig`.

5.7.6 Rozhraní SLIP a PPP

Ačkoliv jsou spojení typu SLIP nebo PPP pouze jednoduchými spojeními typu point-to-point, podobně jako u spojení typu PLIP je vhodné v souvislosti s nimi uvést některé informace. Uskutečnění spojení typu SLIP obvykle vyžaduje zavolání vzdáleného počítače pomocí modemu a nastavení sériové linky do režimu SLIP. Podobně se používá i protokol PPP. Nástroje potřebné pro konfiguraci spojení SLIP a PPP budou popsány v kapitolách 7 a 8.

[7] Avšak z důvodů opatrnosti byste měli konfigurovat spojení PLIP nebo SLIP až poté, co máte kompletně nastaveny ethernetové položky ve směrovací tabulce. V opačném případě by u některých starších jader operačního systému mohlo vaše síťové směrování skončit na spojení point-to-point.

5.7.7 Fiktivní rozhraní

Fiktivní rozhraní je skutečně trošku exotické, nicméně je docela užitečné. Jeho hlavní význam souvisí se samostatnými hostiteli a počítači, jejichž jediné síťové IP-spojení je připojení pomocí modemu. Později se ve skutečnosti začalo mnohem více prosazovat i u samostatných hostitelů.

Dilema u samostatných hostitelů spočívá v tom, že mají aktivní pouze jediné síťové zařízení, a to konkrétně zpětnovazebné zařízení. To má obvykle přidělenu adresu **127.0.0.1**. Nicméně v určitých situacích potřebujete posílat data na oficiální IP-adresu místního hostitele. Například, uvažujme laptop **vlite**, jenž byl během trvání tohoto příkladu odpojen od všech sítí. Aplikace na laptopu **vlite** může chtít poslat nějaká data na vlastního hostitele **vlite**. Prohledá-li na počítači **vlite** soubor `/etc/hosts/`, obdrží IP-adresu **191.72.1.65**, tudíž se aplikace pokusí na tuto adresu poslat data. Protože jediným aktivním rozhraním je pouze zpětnovazebné rozhraní, nemůže jádro operačního systému vědět, že se tato adresa ve skutečnosti vztahuje na něj! Následně jádro operačního systému zničí datagram a vrátí aplikaci chybovou hlášku.

V tomto bodě vstupuje do hry fiktivní rozhraní. Vyřeší dilema tím, že bude sloužit jako pozměněné zpětnovazebné rozhraní. V případě laptopu **vlite** byste mu měli přidělit adresu **191.72.1.65** a přidat směrování hostitele na tuto adresu. Každý datagram pro adresu **191.72.1.65** pak bude doručen lokálně. Správné volání vypadá následovně:

```
# ifconfig dummy vlite
# route add vlite
```


5.8 Vše o příkazu ifconfig

Příkaz `ifconfig` má mnohem více parametrů, než jsme si zatím řekli. Jeho volání v klasické podobě vypadá následovně:

```
ifconfig interface [[-net|-host] address [parameters]]
```

Parametr `interface` představuje název rozhraní, parametr `address` představuje IP-adresu přidělenou danému rozhraní. Může to být buď IP-adresa v tečkové notaci, nebo název, který příkaz `ifconfig` vyhledá v souborech `/etc/hosts` a `/etc/networks`. Volby `-net` a `-host` přinutí příkaz `ifconfig`, aby považoval adresu za číslo sítě, resp. za adresu hostitele.

Je-li příkaz `ifconfig` vyvolán pouze s názvem rozhraní, zobrazí konfiguraci příslušného rozhraní. Je-li spuštěn bez parametrů, zobrazí všechny již dříve nakonfigurovaná rozhraní; volba `-a` donutí příkaz zobrazit i neaktivní rozhraní. Vzorové vyvolání příkazu `ifconfig` pro ethernetové rozhraní *eth0* by mohlo vypadat asi takto:

```
# ifconfig eth0
eth0   Link encap 10Mbps Ehternet  HWaddr 00:00:C0:90:B3:42
       inet addr 191.72.1.2 Bcast 191.72.1.255 Mask 255.255.255.0
       UP BROADCAST RUNNING MTU 1500 Metric 0
       RX packets 3136 errors 217 dropped 7 overrun 26
       TX packets 1752 errors 25 dropped 0 overrun 0
```

Pole `MTU` a `Metric` ukazují aktuální MTU a metrickou hodnotu pro dané rozhraní. Metrickou hodnotu tradičně používají některé operační systémy k výpočtu váhy daného směrování. Linux zatím tuto hodnotu nevyužívá, ale definuje ji z důvodu kompatibility.

Řádky RX a TX ukazují, kolik paketů bylo bezchybně přijato nebo vysláno, ke kolika chybám došlo, kolik paketů bylo zahozeno pravděpodobně z důvodu nedostatku paměti a kolik paketů se ztratilo kvůli přetížení. K přetížení přijímače obvykle dojde v případě, kdy pakety přicházejí rychleji, než stačí jádro operačního systému obsloužit poslední přerušení. Hodnoty příznaků zobrazené příkazem `ifconfig` zhruba odpovídají názvům jeho voleb na příkazovém řádku; budou vysvětleny dále.

Následuje seznam parametrů, které rozeznává příkaz `ifconfig`. Odpovídající názvy symbolů jsou uvedeny v kulatých závorkách. Volby, které zapínají určitou volbu, umožňují i její vypnutí. Toho dosáhnete vložením symbolu mínus (-) před název příslušné volby.

`up` — Označí rozhraní jako „aktivní“, tj. přístupné pro IP-vrstvu. Tato volba předpokládá, že je spolu s ní uvedena na příkazovém řádku i adresa `address`. Lze ji také použít k opětovnému povolení rozhraní, které bylo dočasně zakázáno pomocí volby `down`.

(Této volbě odpovídají symboly UP a RUNNING.)

`down` — Označí rozhraní jako „neaktivní“, tj. nepřístupné pro IP-vrstvu. Tato volba efektivně znemožní jakýkoliv IP-provoz přes dané rozhraní. Všimněte si, že tato volba automaticky nesmaže všechna směrovací data, která používají dané rozhraní. Pokud toto rozhraní trvale znepřístupníte, měli byste tato směrovací data vymazat a je-li to možné, doplnit je o alternativní směrování.

`netmask` *mask* — Tato volba přidělí masku podsítě, kterou bude rozhraní používat. Může být zadána jako 32bitové hexadecimální číslo s předponou `0x` nebo jako čísla desítkové soustavy oddělená tečkou.

`pointopoint` *address*

Tato volba se používá u spojení IP typu point-to-point, které vyžaduje pouze dva hostitele. Tato volba je nutná například při konfiguraci rozhraní SLIP nebo PLIP.

(Byla-li nastavena adresa point-to-point, zobrazí příkaz `ifconfig` symbol `POINTOPOINT`.)

`broadcast` *address*

Vysílací (broadcast) adresa je obvykle vytvořena z čísla sítě nastavením všech bitů části hostitele. Některé implementace protokolu IP používají odlišné schéma; tato volba je zde proto, aby se vysílací adresa přizpůsobila těmto podivným prostředím.

(Je-li nastavena vysílací adresa, zobrazí příkaz `ifconfig` symbol `BROADCAST`.)

`metric` *number* Tato volba slouží k přidělení metrické hodnoty položce směrovací tabulky vytvořené pro dané rozhraní. Tuto metriku používá směrovací informační protokol (RIP) k vytvoření směrovacích tabulek sítí.[8] Implicitní hodnota metriky používaná příkazem `ifconfig` je rovna nule. Pokud nepoužíváte démona RIP, je vám tato volba k ničemu; pokud ano, budete jen zřídka potřebovat tuto hodnotu měnit.

`mtu` *bytes* Tato volba nastavuje tzv. maximální přenosovou jednotku (Maximal Transmission Unit – MTU), která představuje maximální počet oktetů, o něž se může rozhraní postarat při jedné transakci. U sítí typu Ethernet je hodnota MTU implicitně nastavena na 1 500; u rozhraní typu SLIP je MTU nastavena na hodnotu 296.

`arp` Toto je volba typická pro sítě, jako je Ethernet nebo packet radio. Povoluje použití ARP, protokolu pro rozlišení adres, k detekci fyzických adres hostitelů, kteří jsou připojeni k síti. U těchto sítí je tato volba implicitně povolena.

(Je-li protokol ARP zakázán, zobrazí příkaz `ifconfig` symbol `NOARP`.)

`-arp` Na tomto rozhraní zakáže použití protokolu ARP.

[8] Protokol RIP vybírá na základě „délky“ cesty k danému hostiteli optimální směrování. Délku zjistí sečtením metrických hodnot každého spojení mezi hostiteli. Implicitně má skok hodnotu 1, ale ve skutečnosti to může být libovolné celé číslo menší než 16. (Délka směrování 16 odpovídá nekonečné vzdálenosti. Taková směrování jsou považována za nepoužitelná.) Parametr metriky nastavuje váhu skoku, která je potom vysílána směrovacím démonem.

`promisc` Přepne rozhraní do „promiskuitního“ módu. U většiny sítí to bude znamenat, že dané rozhraní bude přijímat všechny pakety, bez ohledu na to, zda byly určeny pro jiného hostitele či nikoliv. Tato volba umožní analyzovat síťový provoz pomocí filtrování paketů (tzv. *sledování Ethernetu*). Je to dobrý způsob, jak vychytat síťové problémy, které by jinak bylo těžké vystopovat.

Na druhou stranu umožní tato volba útočníkům kromě jiných věcí i zjištění potřebných hesel z vašeho síťového provozu. Jednou z ochran proti tomuto typu útoku je všeobecné zakázání pouhého zastrčení svého počítače do vaší ethernetové zásuvky. Další možností je používat bezpečné ověřovací protokoly, jako například Kerberos, nebo přihlašovací balík SRA.[9*]

(Této volbě odpovídá symbol `PROMISC`.)

`-promisc` Tato volba vypne promiskuitní mód.

`allmulti` Multicast-adresy jsou určitým druhem vysílání dat skupině hostitelů, kteří nutně nemusí ležet v jedné podsíti

(Tato volba odpovídá symbolu `ALLMULTI`.)

`-allmulti` Tato volba vypne multicast-adresy.

5.9 Kontrola pomocí příkazu netstat

Nyní se podíváme na užitečný nástroj pro kontrolu konfigurace sítě a síťové aktivity. Nazývá se `netstat` a ve skutečnosti jde spíše o sbírku několika nástrojů shrnutých dohromady. V následujících statích si probereme každou z jeho funkcí.

5.9.1 Zobrazení směrovací tabulky

Spustíte-li příkaz `netstat` s parametrem `-r`, zobrazí se směrovací tabulka jádra operačního systému stejným způsobem, jako tomu bylo u příkazu `route`. Na hostiteli **vstout** se zobrazí:

```
# netstat -nr
Kernel routing table
```

9 Balík SRA můžete získat na serveru **ftp.tamu.edu** z adresáře */pub/sec/TAMU*.

* Poznámka korektora: V poslední době je nejpoužívanějším balíkem pro vzdálené přihlášení program `ssh`. Více informací naleznete na adrese *http://www.ssh.fi*.

```
Destination  Gateway     Genmask          Flags Metric Ref  Use   Iface
127.0.0.     *           255.255.255.255  UH    1      0    50    lo
191.72.1.0   *           255.255.255.0    U     1      0    478   eth0
191.72.2.0   191.72.1.1  255.255.255.0    UGN   1      0    250   eth0
```

Volba `-n` způsobí, že příkaz `netstat` zobrazí adresy jako čísla IP oddělené tečkami, a ne jako symbolické názvy hostitelů a sítí. To je užitečné, chcete-li se vyhnout vyhledávání adres po síti (například na DNS nebo NIS-serveru).

Druhý sloupec výstupu příkazu `netstat` zobrazuje bránu, na kterou ukazují směrovací data. Není-li použita žádná brána, zobrazí se symbol hvězdičky. Sloupec tři ukazuje „všeobecnost" směrování. Je-li zadána IP-adresa, pro kterou se má vyhledat správné směrování, projde jádro systému všechna data směrovací tabulky, na adresu a všeobecnou masku aplikuje bitově orientovanou operaci AND a výsledek porovná s cílem směrování.

Čtvrtý sloupec zobrazuje různé symboly, které popisují dané směrování:

G Směrování používá bránu.

U Používané rozhraní je povolené.

H Daným směrováním může být dosažitelný pouze jediný hostitel. To je například případ dat pro zpětnovazebné rozhraní **127.0.0.1**.

D Tento symbol je zobrazen, pokud byla data vygenerována ICMP-zprávou o přesměrování (viz stať 2.5).

M Tento symbol je zobrazen, pokud byla položka v tabulce změněna ICMP zprávou o přesměrování.

Sloupec `Ref` ve výstupu příkazu `netstat` zobrazuje počet odkazů na dané směrování, to znamená, kolik dalších směrování (například přes brány) spoléhá na přítomnost daného směrování. Poslední dva sloupce zobrazují, kolikrát byla použita směrovací data a dále rozhraní, kterými při doručování procházejí datagramy.

5.9.2 Zobrazování statistik rozhraní

Je-li příkaz `netstat` spuštěn s parametrem `-i`, zobrazí se statistiky pro aktuálně nakonfigurovaná síťová rozhraní. Je-li připojen i parametr `-a`, vypíší se všechna zařízení přítomná v jádru operačního systému, nejenom ta, která již byla nakonfigurována. Na hostiteli **vstout** by výstup příkazu `netstat` vypadal následovně:

```
$ netstat -i
Kernel Interface Table
Iface MTU  Met RX-OK   RX-ERR RX-DRP RX-OVR TX-OK  TX-ERR TX-DRP TX-OVR Flags
lo    0    0   3185    0      0      0      3185   0      0      0      BLRU
eth0  1500 0   972633  17     20     120    628711 217    0      0      BRU
```

Pole `MTU` a `Met` ukazují aktuální hodnoty MTU a metriky daného rozhraní. Sloupce `RX` a `TX` ukazují, kolik paketů bylo bezchybně přijato nebo vysláno (RX-OK/TX-OK), kolik bylo poškozeno (RX-ERR/TX-ERR), kolik bylo zahozeno (RX-DRP/TX-DRP) a kolik se ztratilo z důvodu přetížení (RX-OVR/TX-OVR).

Poslední sloupec zobrazuje symboly nastavené u daného zařízení. Jsou to jednoznakové ekvivalenty dlouhých názvů příznaků, které se vypisují při zobrazení konfigurace rozhraní pomocí příkazu `ifconfig`.

B Byla nastavena vysílací adresa.

L Dané rozhraní je zpětnovazebné rozhraní.

M Jsou přijímány všechny pakety (promiskuitní mód).

O Pro dané rozhraní je vypnut protokol ARP.

P Toto spojení je typu point-to-point.

R Rozhraní právě běží.

U Rozhraní je povoleno.

5.9.3 Zobrazení propojení

Příkaz `netstat` podporuje skupinu voleb pro zobrazení aktivních nebo pasivních socketů. Volby `-t`, `-u`, `-w` a `-x` ukazují aktivní TCP, UDP, RAW nebo unixová socketová spojení. Pokud k nim doplníte i parametr `-a`, budou zobrazeny i sockety čekající na spojení (například při naslouchání). Tato kombinace parametrů vám poskytne úplný výpis všech serverů, které právě běží ve vašem systému.

Při použití příkazu `netstat -ta` se na hostiteli **vlager** zobrazí následující výpis:

```
$ netstat -ta
Active Internet connections
Proto Recv-Q Send-Q Local Address  Foreign Address    (State)
tcp    0      0      *:domain       *:*              LISTEN
tcp    0      0      *:time         *:*              LISTEN
tcp    0      0      *:smtp         *:*              LISTEN
tcp    0      0      vlager:smtp    vstout:1040      ESTABLISHED
tcp    0      0      *:telnet       *:*              LISTEN
tcp    0      0      localhost:1046 vbardolino:telnet ESTABLISHED
tcp    0      0      *:chargen      *:*              LISTEN
tcp    0      0      *:daytime      *:*              LISTEN
tcp    0      0      *:discard      *:*              LISTEN
tcp    0      0      *:echo         *:*              LISTEN
tcp    0      0      *:shell        *:*              LISTEN
tcp    0      0      *:login        *:*              LISTEN
```

Tento výpis ukazuje, že většina serverů čeká na příchozí spojení. Nicméně čtvrtý řádek ukazuje příchozí spojení typu SMTP z hostitele **vstout** a šestý řádek vám sděluje, že existuje výstupní spojení typu telnet s hostitelem **vbardolino**.[10]

Pokud bychom použili pouze argument `-a`, zobrazí se všechny sockety ze všech rodin.

5.10 Kontrola tabulek ARP

V určitých situacích je vhodné si prohlédnout, případně změnit obsah tabulek ARP jádra systému. Například máte-li podezření, že příčinou občasných síťových problémů je duplicitní internetová adresa. Pro takovéto situace byl vytvořen nástroj `arp`. Jeho volby příkazové řádky jsou následující:

```
arp [-v] [-t hwtype] -a [hostname]
arp [-v] [-t hwtype] -s hostname hwaddr
arp [-v] -d hostname [hostname...]
```

Všechny argumenty názvu hostitele `hostname` jsou buďto symbolické názvy hostitelů, nebo IP-adresy respektující tečkovou notaci.

[10] Zda je dané spojení výstupní, zjistíte podle jeho čísla portu. Číslo portu bude pro *volajícího* hostitele vždy přirozené, zatímco u volaného hostitele se použije známý obslužný port, pro nějž příkaz `netstat` používá symbolický název nacházející se v souboru `/etc/services`.

První typ volání příkazu `arp` zobrazí pro danou IP-adresu nebo pro daného hostitele položku ARP. V případě, že nebyl zadán název hostitele `hostname`, zobrazí se položky ARP pro všechny známé hostitele. Například vyvolání příkazu `arp` na hostiteli **vlager** může vypadat takto:

```
# arp -a
IP address          HW type                 HW address
191.72.1.3          10Mbps Ethernet         00:00:C0:5A:42:C1
191.72.1.2          10Mbps Ethernet         00:00:C0:90:B3:42
191.72.2.4          10Mbps Ethernet         00:00:C0:04:69:AA
```

Zobrazí se ethernetové adresy hostitelů **vlager**, **vstout** a **vale**.

Pomocí volby `-t` můžete omezit výpis pouze na zadaný typ hardwaru. Jím může být *ether*, *ax25* nebo *pronet*, což odpovídá 10Mbps Ethernetu, AMPR AX.25, resp. zařízení IEEE 802.5 Token Ring.

Parametr `-s` slouží k permanentnímu přidání ethernetové adresy názvu hostitele `hostname` do tabulek ARP. Argument `hwaddr` určuje hardwarovou adresu, u níž se implicitně předpokládá, že jde o ethernetovou adresu určenou šesti hexadecimálními bajty oddělenými dvojtečkou. Za pomoci volby `-t` můžete také nastavit hardwarové adresy pro jiné typy hardwaru.

Někdy bude nutné manuální doplnění IP-adresy do tabulky ARP, například když z nějakých příčin selžou na vzdáleném hostiteli dotazy ARP, k čemuž může dojít, je-li na vzdáleném hostiteli chybný ovladač ARP, nebo když v síti existuje další hostitel, který se chybně identifikuje IP-adresou vzdáleného hostitele. Zapsání IP-adresy napevno do tabulky ARP je opatření (velice drastické), kterým se chráníte proti hostitelům z vašeho Ethernetu, jež se vydávají za někoho jiného.

Použijete-li při spuštění příkazu `arp` parametr `-d`, smažou se všechna data ARP, která se vztahují k danému hostiteli. Pomocí tohoto parametru můžete rozhraní přinutit k tomu, aby se znovu pokusilo pomocí dotazu získat ethernetovou adresu odpovídající dané IP-adrese. To je užitečné v případě, kdy špatně nakonfigurovaný systém vysílá špatné informace ARP (samozřejmě předtím musíte chybného hostitele znovu zkonfigurovat).

Volba `-s` slouží k implementaci techniky *proxy* ARP. Je to speciální technika, kdy se hostitel, například **gate**, chová vůči dalšímu hostiteli **fnord** jako brána a předstírá, že se obě adresy vztahují ke stejnému hostiteli, konkrétně k hostiteli **gate**. Provede to tak, že zveřejní ARP položku hostitele **fnord**, která bude ukazovat na své vlastní ethernetové rozhraní. Když nyní nějaký hostitel pošle dotaz ARP na hostitele **fnord**, hostitel **gate** vrátí odpověď, která bude obsahovat jeho vlastní ethernetovou adresu. Potom pošle dotazující se hostitel všechny datagramy na hostitele **gate**, který je následně zašle hostiteli **fnord**.

Tyto záměny jsou nutné například v případě, kdy chcete přistupovat k hostiteli **fnord** z počítače s operačním systémem DOS, který nemá zcela korektní implementaci TCP s dobrým směrováním. Při použití techniky proxy ARP se bude hostitel jevit počítači s operačním systémem DOS jako by byl **fnord** v místní podsíti, takže počítač s operačním systémem DOS nemusí vůbec umět směrovat pomocí brány.

Další velice užitečnou aplikací techniky proxy ARP je případ, kdy se jeden z vašich hostitelů chová jako brána vůči nějakému jinému hostiteli jen dočasně, například při připojení pomocí modemu. V předešlém příkladu jsme se již setkali s laptopem **vlite**, který je občas spojován s bránou **vlager** spojením typu PLIP. Samozřejmě, že tento způsob bude fungovat pouze v případě, kdy je adresa hostitele, na kterém chcete provozovat techniku proxy ARP, ve stejné podsíti IP jako vaše brána. Například hostitel **vstout** by mohl používat techniku proxy ARP pro libovolného hostitele sítě pivovaru (**191.72.1.0**), ale nikdy pro hostitele z podsítě vinárny (**191.72.2.0**).

Správné vyvolání příkazu `arp`, kdy bude hostiteli **fnord** poskytnuta technika proxy ARP, je uvedeno níže; samozřejmě, že předaná ethernetová adresa musí odpovídat hostiteli **gate**.

```
# arp -s fnord 00:00:c0:a1:42:e0 pub
```

Položku proxy ARP můžete opět odstranit následujícím způsobem:

```
# arp -d fnord
```

5.11 Budoucnost

Síťové služby Linuxu se stále vyvíjí. Velké změny v hladině jádra operačního systému s sebou přinesou pružné konfigurační schéma, které vám umožní konfigurovat síťové zařízení i za provozu. Například příkaz `ifconfig` bude akceptovat argumenty nastavující IRQ a kanál DMA.

V blízké budoucnosti se počítá s doplněním příznaku `mtu`, který nastaví pro konkrétní směrování maximální přenosovou jednotku. Tato MTU specifická pro dané směrování potlačí MTU specifikované pro dané rozhraní. Tuto volbu budete obvykle používat u směrování přes bránu, kde bude spojení mezi bránou a cílovým hostitelem vyžadovat velmi malou hodnotu MTU. Dejme tomu, že je třeba hostitel **wonderer** připojen k bráně **vlager** spojením typu SLIP. Když budete posílat data z hostitele **vstout** hostiteli **wonderer**, bude síťová vrstva hostitele **wonderer** používat pakety o velikosti až 1 500 bajtů, protože jsou tyto pakety posílány po Ethernetu. Na druhé straně však pracuje spojení typu SLIP s MTU o velikosti 296 bajtů, takže síťová hladina hostitele **vlager** musí rozdělit IP-pakety na menší části, které by nepřesa-

hovaly 296 bajtů. Kdyby bylo naopak možné nastavit směrování na hostiteli **vstout** tak, aby se od začátku používalo MTU o velikosti 296 bajtů, pak bychom se této poměrně náročné fragmentaci vyhnuli:

```
# route add wanderer gw vlager mtu 296
```

Všimněte si, že volba `mtu` vám umožní selektivně vrátit účinek opatření, které chápe podsítě jako místní (‘Subnets Are Local‘ Policy – SNARL). Toto opatření je konfigurační volbou jádra operačního systému a je popsáno v kapitole 3.

Kapitola

6

Konfigurace resolveru a jmenných služeb

Ve 2. kapitole jsme si řekli, že sítě na bázi protokolu TCP/IP mohou používat různá schémata konverze názvů na adresy. Nejjednodušším způsobem, který však nevyužívá výhody rozdělení jmenného prostoru do jednotlivých zón, je tabulka hostitelů, která je uložena v souboru `/etc/hosts`. Tento způsob je vhodný pouze pro malé lokální sítě, které spravuje jediný správce a které nepoužívají žádnou IP-komunikaci s okolním světem. Formát souboru `hosts` byl podrobně popsán v kapitole 5.

Další možností je použití služby BIND – Berkeley Internet Name Domain Service – ta přiděluje názvy hostitelů jednotlivým IP-adresám. Konfigurace služby BIND je skutečným oříškem, ale jakmile ji jednou zvládnete, bude pro vás začlenění případných změn v síťové topologii velmi snadné. V systému Linux, stejně jako u dalších unixových systémů, poskytuje jmenné služby program `named`. Program `named` načte při startu několik hlavních souborů do své vyrovnávací paměti a dále bude čekat na dotazy od vzdálených nebo místních uživatelských procesů. Existuje několik způsobů, jak nastavit službu BIND a zdaleka ne všechny vyžadují spuštěný jmenný server na každém hostiteli.

V této kapitole by bylo možné uvést mnohem více informací, než jen hrubý náčrt způsobu, jakým lze provozovat jmenný server. Hodláte-li používat službu BIND v prostředí s většími sítěmi než jen LAN a pravděpodobně i s připojením na Internet, pak byste si měli sehnat nějakou kvalitní knihu pojednávající o službě BIND, například knížku „DNS and BIND“ od Cricketa Liua (viz [AlbitzLiu92]). Aktuální informace najdete také v poznámkách vydávaných společně se zdrojovými soubory služby BIND. Otázkám systému DNS se věnuje i konference nazvaná **comp.protocols.tcp-ip.domains**.

6.1 Knihovna resolveru

Hovoříme-li o „resolveru“, nemáme namysli žádnou speciální aplikaci, ale spíše odkazy do *knihovny resolveru*, což je skupina funkcí nacházející se ve standardní knihovně jazyka C. Centrálními rutinami jsou procedury *gethostbyname(2)* a *gethostbyaddr(2)*, které umí vyhledat všechny IP-adresy náležející danému hostiteli, a versa vice. Mohou být nastaveny tak, aby využívaly soubor `hosts`, aby se dotazovaly na počet jmenných serverů nebo aby používaly databázi `hosts` služby NIS (síťové informační služby). Další aplikace, jako je například `smail`, mohou pro tyto účely obsahovat různé ovladače, které však vyžadují speciální péči.

6.1.1 Soubor host.conf

Centrálním souborem, který řídí nastavení resolveru, je soubor `host.conf`. Nachází se v adresáři `/etc` a sděluje resolveru, jakou by měl použít službu a v jakém pořadí.

Jednotlivé volby musí být v souboru `host.conf` uvedeny na samostatných řádcích. Pole mohou být oddělena bílými mezerami (což jsou mezery nebo tabulátory). Symbol # označuje komentář, který končí u dalšího znaku nové řádky.

K dispozici jsou následující volby:

order Tato volba určuje pořadí, ve kterém budou zkoušeny jednotlivé služby resolveru. Přípustné volby jsou *bind* pro použití dotazů na jmenný server, *hosts* pro vyhledávání v souboru `/etc/hosts` a *nis* pro vyhledávání pomocí služby NIS. Může být zadána buď jedna nebo více voleb. Pořadí, ve kterém jsou tyto volby uvedeny, bude určovat pořadí, v kterém budou zkoušeny s nimi související služby.

multi Argumenty této volby jsou *on* nebo *off*. Určuje, zda může mít hostitel v souboru `/etc/hosts` několik IP-adres. Adresy tohoto typu se někdy označují také jako multihomed adresy. U dotazů systému DNS nebo NIS nemá tento přepínač žádný význam.

nospoof V minulé kapitole jsme si řekli, že systém DNS umí najít název hostitele náležející dané IP-adrese. K tomu používá doménu **in-addr.arpa**. Snahy jmenných serverů navrátit chybný název hostitele se označují jako tzv. „*spoofing*“. Obrana proti možným chybám spočívá v konfiguraci resolveru takovým způsobem, aby zjišťoval, zda je původní adresa skutečně spjatá se získaným názvem hostitele. Pokud nikoliv, je název odmítnut a vrácena chybová zpráva. Toto chování se zapíná pomocí volby *nospoof on*.

alert Tato volba má argumenty *on* nebo *off*. Je-li zapnutá, pak veškeré pokusy o spoofing (viz výše) budou zapsány do souboru *syslog*.

trim Tato volba má jako argument název domény, která bude před vyhledáváním z názvů hostitelů odstraněna. Volba je užitečná u těch položek v souboru *hosts*, u kterých jste zadali pouze názvy hostitelů bez místní domény. Při vyhledání lokálního hostitele, jenž má k sobě připojenu místní doménu, bude tato doména odstraněna, takže vyhledání v souboru `/etc/hosts` bude úspěšné.

Volby *trim* lze použít vícekrát, takže lze vašeho hostitele považovat za místního hostitele vůči několika doménám.

Následuje vzorový soubor pro bránu **vlager**:

```
# /etc/host.conf
# Používáme named, NIS (zatím) ne
order   bind hosts
# Jeden hostitel může mít více adres
multi   on
# Ochrana proti spoofingu
nospoof on
# Odstraňuji lokální doménu
trim    vbrew.com.
```

6.1.2 Proměnné pro prostředí resolveru

Nastavení v souboru `host.conf` lze potlačit pomocí několika proměnných prostředí resolveru.

Následuje výpis těchto proměnných:

`RESOLV_HOST_CONF`
Určuje soubor, který bude načten místo souboru `/etc/host.conf`.

`RESOLV_SERV_ORDER`
Potlačí volbu *order* v souboru *host.conf*. Službami mohou být *hosts*, *bind* nebo *nis*. Odděleny mohou být mezerou, čárkou nebo středníkem.

`RESOLV_SPOOF_CHECK`
Určí opatření, která budou použita proti spoofingu. Při hodnotě *off* je kontrola úplně vypnutá. Hodnoty *warn* a *warn off* kontrolují spoofing, avšak je zapnuto respektive vypnuto zapisování chyb. Hodnota * bude kontrolovat spoofing, ale rozsah zaznamenávání chyb ponechá nastavený podle souboru `host.conf`.

`RESOLV_MULTI`
K potlačení voleb *multi* v souboru `host.conf` lze použít hodnoty *on* nebo *off*.

`RESOLV_OVERRIDE_TRIM_DOMAINS`
Tato proměnná určuje seznam domén, který potlačí domény uvedené v souboru *host.conf*.

`RESOLV_ADD_TRIM_DOMAINS`
Tato proměnná specifikuje seznam domén určených k odstranění. Tyto domény budou přidány k doménám, které jsou uvedeny v souboru `host.conf`.

6.1.3 Konfigurace vyhledávání jmenného serveru – resolv.conf

Nakonfigurujete-li knihovnu resolveru tak, aby k vyhledávání hostitelů používala jmennou službu BIND, musíte jí sdělit, jaké má používat jmenné servery. K tomuto účelu se používá speciální soubor s názvem `resolv.conf`. Pokud tento soubor neexistuje nebo je prázdný, bude resolver předpokládat, že se jmenný server nachází na vašem místním hostiteli.

Provozujete-li jmenný server na svém místním hostiteli, musíte ho nastavit samostatně, což si vysvětlíme v dalších statích. Pokud se nacházíte v lokální síti a máte možnost používat existující jmenný server, pak bude tento způsob vždy preferován.

Nejdůležitější volbou v souboru `resolv.conf` je volba *nameserver*, která obsahuje IP-adresu používaného jmenného serveru. Zadáte-li několikanásobným uvedením volby *nameserver* několik jmenných serverů, pak budou tyto jmenné servery zkoušeny v uvedeném pořadí. Z toho důvodu byste měli na prvním místě uvést nejspolehlivější jmenný server. V současné době jsou podporovány až tři jmenné servery.

Není-li uvedena žádna volba *nameserver*, pokusí se resolver spojit se jmenným serverem na místním hostiteli.

Další dvě volby *domain* a *search* ovládají implicitní domény, které budou připojeny k názvu hostitele v případě, že se službě BIND nepodaří při prvním dotazu nalézt příslušný název hostitele. Volba *search* určuje seznam zkoušených názvů domén. Jednotlivé položky v seznamu jsou od sebe odděleny mezerami nebo tabulátory.

Pokud žádnou volbu *search* neuvedete, bude z místního názvu domény sestaven implicitní vyhledávací seznam tím způsobem, že se použije název domény plus všechny nadřazené názvy domén až po kořenovou úroveň. Místní název domény je možné zadat pomocí volby *domain*; není-li tato volba uvedena, pak resolver získá tento název pomocí systémového volání procedury *getdomainname(2)*.

Zdá-li se vám to zmatené, pak se podívejte na následující příklad souboru `resolv.conf`, který používá společnost Virtual Brewery:

```
# /etc/resolv.conf
# Naše doména
domain          vbrew.com
#
# Hostitel vlager je centrálním jmenným serverem:
nameserver      191.72.1.1
```

Při rozkládání názvu **vale** by měl resolver nejprve vyhledat název **vale**. Neuspěje-li, měl by pokračovat vyhledáváním názvů **vale.vbrew.com** a **vale.com**.

6.1.4 Robustnost resolveru

Pokud používáte menší lokální síť v rámci rozsáhlé sítě, měli byste rozhodně používat centrální jmenné servery, jsou-li tyto k dispozici. Výhoda tohoto způsobu spočívá v tom, že centrální jmenné servery si vytvoří velkou vyrovnávací paměť, protože na ně budou směrovány veškeré dotazy. Toto schéma má ale i nevýhody: pokud vám například nedávno zničil oheň kabel na páteřní univerzitní síti, nebude v místním oddělení možné provozovat lokální síť, protože resolver se nebude schopen spojit se žádným jmenným serverem. Nebudete se moci přihlásit k žádnému X-terminálu, nebude fungovat tisk atd.

I když není příliš běžné, aby začala hořet páteř univerzitní sítě, budete asi chtít proti takovýmto případům učinit určitá opatření.

Jednou z možností je nastavit místní jmenný server tak, aby hledal názvy hostitelů z vaší místní domény a všechny ostatní dotazy na další názvy hostitelů předával na hlavní servery. Samozřejmě, že toto schéma bude fungovat pouze v případě, že provozujete svou vlastní doménu.

Máte i druhou možnost – v souboru `/etc/hosts` udržovat záložní tabulku hostitelů vaší domény nebo lokální sítě. Následně byste měli do souboru `/etc/host.conf` přidat volby „*order bind hosts*“, aby se resolver v případě nefunkčního centrálního jmenného serveru vrátil zpět k souboru hostitelů.

6.2 Provozování programu named

Program, který na většině unixových počítačích poskytuje doménové jmenné služby, se obvykle jmenuje `named` ([neimdi:]). Jedná se o serverový program původně vyvinutý pro operační systém BSD, který poskytuje jmenné služby klientům a případně i dalším jmenným serverům. Zdá se, že v současnosti se na většině instalacích Linuxu používá verze BIND-4.8.3. Novější verze BIND-4.9.3 existuje momentálně ve verzi beta a již brzy by se měla stát součástí Linuxu.

Tato stať vyžaduje určité znalosti způsobu, jakým funguje doménový jmenný systém. Budou-li pro vás následující diskuse tak trochu španělskou vesnicí, mě-li byste si znovu přečíst 2. kapitolu, kde najdete více informací o základech systému DNS.

Program `named` je obvykle spuštěn při zavádění systému a běží tak dlouho, dokud počítač nevypnete. Informace získává z konfiguračního souboru `/etc/named.boot` a z mnoha dalších souborů, které obsahují data týkající se mapování názvů domén na IP-adresy atp. Posledně zmiňované soubory se nazývají *soubory zón*. Jejich sémantika a formát bude vysvětlen v následující stati.

Program `named` spustíte v příkazové řádce zadáním:

```
# /usr/sbin/named
```

Program `named` načte konfigurační soubor `named.boot` a všechny další v něm uvedené soubory zón. Své identifikační číslo procesu zapíše ve formátu ASCII do souboru `/var/run/named.pid` a je-li to nutné, načte soubory zón z primárních serverů a spustí na portu číslo 53 odposlouchávání DNS dotazů.[1]

6.2.1 Soubor named.boot

Soubor `named.boot` je zpravidla velmi malý a obsahuje pouze ukazatele na hlavní soubory s informacemi o zónách a ukazatele na další jmenné servery. V tomto zaváděcím souboru začínají komentáře středníkem a končí u dalšího znaku nové řádky. Dříve, než si probereme formát souboru `named.boot` podrobněji, podíváme se na vzorový soubor pro bránu **vlager**, který vidíte na obrázku 6.1.[2]

[1] Na FTP serverech se nachází několik druhů binárních souborů programu `named`, z nichž každý se nastavuje nepatrně odlišným způsobem. Některé z nich mají svůj soubor pid uložen v adresáři `/etc`, jiné ho ukládají do adresářů `/tmp` nebo `/var/tmp`.

[2] Všimněte si, že názvy domén v tomto příkladu nejsou uváděny *včetně* postfixové tečky. Starší verze programu `named` chápaly postfixové tečky jako chybu a příslušnou řádku v tichosti ignorovaly. Verze programu BIND-4.9.3 by snad měla mít tuto chybu opravenu.

```
;
; /etc/named.boot pro vlager.vbrew.com
;
directory      /var/named
;
;             doména                        soubor
;-------------------------------------------------
cache         .                             named.ca
primary       vbrew.com                     named.hosts
primary       0.0.127.in-addr.arpa          named.local
primary       72.191.in-addr.arpa           named.rev
```

Obrázek 6.1
Soubor *named.boot* pro hostitele **vlager**

Příkazy `cache` a `primary`, použité v tomto příkladu, načítají do programu `named` informace. Tyto informace jsou převzaty z hlavních souborů zadaných v druhém argumentu. Obsahují textové verze položek zdrojových záznamů systému DNS, které si popíšeme dále.

V tomto příkladu jsme nakonfigurovali program `named` jako primární jmenný server pro tři domény, což naznačují tři příkazy `primary` na konci tohoto souboru. První z těchto tří řádek například říká programu `named`, aby se choval jako primární server pro adresu **vbrew.com** a aby načetl data ze souboru `named.hosts`. Klíčové slovo `directory` udává, že všechny soubory zón jsou umístěny v adresáři `/var/named`.

Volba `cache` je speciální a rozhodně by měla být přítomna na všech počítačích provozujících jmenný server. Má dvojí funkci: přikazuje programu `named`, aby povolil vyrovnávací paměť a načítá ze zadaného souboru vyrovnávací paměti (v našem příkladu je to soubor `named.ca`) kořenové jmenné servery. Ke kořenovým jmenným serverům se vrátíme později.

Následuje seznam nejdůležitějších voleb, které můžete použít v souboru `named.boot`:

`directory` Tato volba určuje adresář, ve kterém budou umístěny soubory zón. Názvy souborů mohou být zadávány relativně vůči tomuto adresáři. Několikanásobným použitím volby `directory` lze zadat i více adresářů. Podle standardů souborového systému Linuxu by tímto adresářem měl být adresář `/var/named`.

`primary` Tato volba používá jako argumenty **název domény** a **název souboru** a deklaruje autoritativní jmenný server pro danou doménu. Protože jde o primární server, načte program `named` informace o zónách z příslušného hlavního souboru.

Obecně bude v souboru `named.boot` vždy minimálně jedna položka s volbou `primary`, a tou bude zpětné mapování sítě **127.0.0.0**, což je místní zpětnovazebná síť.

`secondary` Tato volba používá jako argumenty **název domény**, **seznam adres** a **název souboru**. Pro danou doménu určuje místní server, který se stane sekundárním jmenným serverem.

Sekundární server také uchovává důležité údaje o doméně. Nezískává je však ze souborů, ale snaží se je stáhnout z primárního serveru. Proto musí být v seznamu adres příkazu `named` uveden minimálně jeden primární server. Místní server bude kontaktovat každý server z daného seznamu, dokud se mu nepodaří úspěšně přenést databázi zóny, která bude potom uložena jako záložní soubor pod názvem, který byl uveden jako třetí argument. Neodpovídá-li ani jeden z primárních serverů, použijí se data získaná ze záložních souborů.

Příkaz `named` se potom pokusí v pravidelných intervalech obnovovat data zóny. Tato problematika je vysvětlena dále společně se zdrojovými záznamy typu SOA.

`cache` Tato volba má jako argumenty **doménu** a **název souboru**. Tento soubor obsahuje seznam záznamů ukazujících na kořenové jmenné servery. Jsou rozlišovány pouze záznamy typů NS a A. Argumentem **doména** je obvykle název kořenové domény „.“.

Tyto informace jsou pro program `named` rozhodující: Pokud v souboru `named.boot` neexistuje volba `cache`, nebude program `named` vytvářet místní vyrovnávací paměť. To způsobí výrazné snížení výkonu a zvýšení zatížení sítě v případě, že se dotazovaný server nenachází v místní síti. Kromě toho se nebude moci program `named` spojit s žádným kořenovým jmenným serverem, a tak nebude moci rozložit žádné adresy kromě těch, pro které je správcem. Výjimkou z tohoto pravidla je použití tzv. forwardujících serverů (srov. s níže uvedenou volbou `forwarders`).

`forwarders` Tato volba má jako argument **seznam adres**. IP-adresy v seznamu určují seznam jmenných serverů, kterých se může program `named` dotazovat v případě, že se mu nepodaří vyřešit dotaz pomocí své místní vyrovnávací paměti. Jmenné servery jsou v příslušném pořadí neustále dotazovány, dokud některý z nich neodpoví na dotaz.

`slave` Tento příkaz označí jmenný server jako *řízený* server. To znamená, že nikdy nebude sám provádět rekurzivní dotazy, ale bude je posílat na servery uvedené ve volbě `forwarders`.

Ještě jsme neuvedli dvě další volby, `sortlist` a `domain`. Kromě toho existují další dvě direktivy, které lze použít uvnitř databázových souborů zón. Jedná se o příkazy *$INCLUDE* a *$ORIGIN*. Protože jsou používány jen velmi zřídka, nebudeme se jimi dále zabývat.

6.2.2 Databázové soubory systému DNS

Hlavní soubory, které využívá program `named`, například soubor `named.hosts`, obsahují vždy nějakou doménu, se kterou jsou sdruženy. Tato doména se nazývá *počátek (origin)*. Doménu tvoří název domény zadaný pomocí příkazů `cache` a `primary`. V rámci hlavního souboru můžete zadávat názvy domén a hostitelů relativně vůči této doméně. Název uvedený v konfiguračním souboru je považován za *absolutní*, pokud končí jednou tečkou, v opačném případě je považován za relativní vůči počátku. Vlastní počátek se může odkazovat sám na sebe za pomoci symbolu „@“.

Všechna data obsažená v hlavním souboru jsou rozdělena do tzv. *zdrojových záznamů (resource record)*, zkráceně záznamy RR. Vytvářejí nejmenší jednotky informace dostupné pomocí systému DNS. Každý zdrojový záznam je určitého typu. Například záznamy typu A mapují název hostitele na IP-adresu a záznam typu CNAME přiřazuje přezdívky hostitele oficiálnímu názvu hostitele. Máte-li zájem o nějaký příklad, podívejte se na obrázek 6.3, který ukazuje hlavní soubor `named.hosts` ve společnosti Virtual Brewery.

Položky zdrojových záznamů v hlavních souborech sdílejí následující společný formát:

```
[domain] [ttl] [class] type rdata
```

Pole jsou navzájem oddělena pomocí mezer nebo tabulátorů. Položka může pokračovat na několika řádcích, pokud před první řádek vložíte levou kulatou závorku a za poslední pole vložíte pravou kulatou závorku. Vše mezi středníkem a novým řádkem bude ignorováno.

`domain` Toto je název domény, ke které se vztahuje daná položka. Není-li uveden žádný název domény, bude se předpokládat, že se záznam RR vztahuje k doméně uvedené v předchozím záznamu RR.

`ttl` Aby bylo po uplynutí určité doby možno donutit resolver k vyřazení určitých informací, je ke každému záznamu RR přirazen tzv. „*čas přežití*“, zkráceně `ttl`. Pole `ttl` udává čas v sekundách, po který budou ještě informace po stažení ze serveru platné. Čas `ttl` je desítkové číslo s maximálně osmi číslicemi.

Nezadáte-li žádný časový údaj `ttl`, bude jeho implicitní hodnota rovna hodnotě pole *minimum* předcházejícího záznamu typu SOA.

`class` Tato volba určuje třídu adresy, například třídu IN pro IP-adresy nebo HS pro objekty z třídy Hesiod. U sítí na bázi protokolu TCP/IP by měla být tato volba rovna IN.

Není-li pole `class` zadáno, použije se třída z předchozího záznamu typu RR.

`type` Tato volba popisuje typ záznamu RR. Nejběžnějšími typy jsou záznamy A, SOA, PTR a NS. V následující stati budeme popisovat různé typy záznamů RR.

`rdata` Tato volba se stará o data sdružená se záznamem RR. Formát tohoto pole závisí na typu záznamu RR. Dále bude tato volba popisována odděleně pro každý typ záznamu RR.

Následuje nekompletní seznam typů záznamů RR, které lze použít v hlavních souborech systému DNS. Existuje sice ještě více typů záznamů, ale zde je nebudeme popisovat. Jsou experimentální, a proto mají jen malé obecné použití.

SOA Tento typ záznamu popisuje správní zónu (SOA znamená „počátek správy – Start of Authority"). Tento záznam signalizuje, že záznamy následující po záznamu RR typu SOA budou obsahovat správní informace o doméně. Každý hlavní soubor obsažený v příkazu *primary* musí pro tuto zónu obsahovat záznam typu SOA. Zdrojová data obsahují následující pole:

origin To je kanonický název hostitele primárního jmenného serveru této domény. Obvykle je zadán jako absolutní název.

contact Toto je e-mailová adresa osoby odpovědné za správu domény, u které je znak ‚@' nahrazen tečkou. Je-li například ve společnosti Virtual Brewery odpovědnou osobou **janet**, potom by toto pole mělo obsahovat adresu *janet.vbrew.com*

serial Toto je číslo verze souboru zóny vyjádřené jednou desítkovou číslicí. Kdykoliv se v souboru zóny změní data, mělo by se toto číslo zvýšit.

Sériová čísla používají sekundární jmenné servery k rozpoznávání změn v informacích o zónách. Aby byly tyto informace aktuální, požadují sekundární servery v určitých intervalech po primárních serverech záznam typu SOA a porovnávají jeho sériové číslo s číslem,

které je obsaženo v záznamu typu SOA uloženém ve vyrovnávací paměti. Změnilo-li se toto číslo, potom sekundární servery přenesou z primárního serveru celou databázi zóny.

`refresh` Tato volba udává interval v sekundách, po který mají sekundární servery čekat, než provedou opětovné zkontrolování záznamu typu SOA s primárním serverem. Opět se jedná o desítkové číslo s maximálně osmi číslicemi.

Síťová topologie se obecně příliš často nemění, takže by toto číslo mělo v rozsáhlejších sítích odpovídat zhruba dnům a v menších sítí by tento interval měl být ještě delší.

`retry` Toto číslo určuje interval, po jehož uplynutí by se měl sekundární server znovu spojit s primárním serverem, když se nepodaří nějaký požadavek nebo aktualizace zónových informací. Tento interval by neměl být příliš malý, jinak by dočasný výpadek serveru nebo nějaký síťový problém mohl způsobit obrovské plýtvání se síťovými zdroji ze strany sekundárních serverů. Vhodnou hodnotou pro tento interval je jedna hodina nebo půl hodiny.

`expire` Tato volba udává čas v sekundách, po jehož uplynutí by měl server skartovat všechna data o zónách, pokud se mu během tohoto intervalu nepodařilo spojit s primárním serverem. Normálně by měla být tato hodnota hodně velká. Craig Hunt ([Hunt92]) doporučuje 42 dnů.

`minimum` Toto je implicitní hodnota času `ttl` pro zdrojové záznamy, které ji nemají explicitně zadanou. Tato volba přikazuje ostatním jmenným serverům, aby po určité předem zadané době zrušily záznamy RR. Nemá však nic společného s časem, po kterém se budou snažit sekundární servery o aktualizaci svých informací.

Hodnota *minimum* by měla být dostatečně vysoká, zejména u sítí typu LAN, u nichž se prakticky nikdy nemění síťová topologie. Vhodné je použít hodnotu týden nebo dokonce měsíc. V případech, kde se jednotlivé záznamy RR často mění, můžete použít vlastní dobu `ttl`.

A — Tento typ záznamu RR přiřazuje IP-adresu k názvu hostitele. Pole zdrojových dat obsahuje adresu respektující tečkovou notaci.

Každý hostitel musí mít pouze jeden záznam typu A. Název hostitele použitý v tomto záznamu typu A je považován za oficiální název hostitele neboli za *kanonický* název hostitele. Všechny další názvy hostitelů jsou přezdívky a pomocí záznamu typu CNAME musí být mapovány na kanonický název hostitele.

NS — Tento typ záznamu RR ukazuje na hlavní jmenný server podřazené zóny. Vysvětlení, proč někdo potřebuje záznamy typu NS, najdete ve stati 2.6. Pole zdrojových dat obsahuje název hostitele jmenného serveru. K nalezení názvu hostitele potřebujeme ještě jeden záznam typu A (někdy se mu také říká *tmelící záznam*), který poskytuje IP-adresu jmenného serveru.

CNAME — Danému hostiteli přiřadí přezdívku, která bude propojena s jeho *kanonickým názvem*. Kanonický název hostitele je takový název, pro který existuje v hlavním souboru záznam typu A; přezdívky jsou s tímto názvem jednoduše spojeny pomocí záznamu typu CNAME, ale jinak nemají žádné další vlastní záznamy.

PTR — Tento typ záznamu slouží ke sdružení názvů v doméně **in-addr.arpa** s názvy hostitelů. Tento záznam se používá ke zpětnému mapování IP-adres na názvy hostitelů. Zadaný název hostitele musí být v kanonickém tvaru.

MX — Tento záznam RR definuje *poštovní server* (mail exchanger) pro danou doménu. Důvody použití poštovních serverů jsou probírány ve stati Směrování pošty v Internetu, kterou najdete ve 13. kapitole. Syntaxe záznamu typu MX je tato:

```
[domain] [ttl] [class] MX preference host
```

Argument `host` přiděluje poštovnímu serveru pro doménu `domain` název hostitele `host`. Každý poštovní server má přiřazen celočíselný argument `preference`. Zprostředkovatel přenosu pošty, který chce doručit poštu do domény `domain`, se bude tak dlouho snažit spojit se všemi hostiteli, kteří mají pro danou doménu uvedený záznam typu MX, dokud neuspěje. Nejprve je kontaktován hostitel s nejnižší hodnotou argumentu `preference`, následně ostatní hostitelé v pořadí, které určuje zvyšující se hodnota argumentu `preference`.

HINFO — Tento typ záznamu poskytuje informace o hardwaru a softwaru daného systému. Jeho syntaxe je:

```
[domain] [ttl] [class] HINFO hardware software
```

Pole `hardware` určuje typ hardwaru, který používá daný hostitel. Pro jeho specifikaci existují určité zvyklosti. Seznam korektních názvů obsahuje specifikace „Assigned Numbers“ (RFC 1340). Pokud toto pole obsahuje nějaké mezery, pak musí být tyto mezery uzavřeny do uvozovek. Pole `software` obsahuje používaný operační systém. Opět by měl být vybrán korektní název ze seznamu „Assigned Numbers“ RFC.*

6.2.3 Sestavení hlavních souborů

Obrázky 6.2, 6.3, 6.4 a 6.5 obsahují vzorové příklady souborů pro jmenný server pivovaru, který je umístěn na hostiteli **vlager**. Vzhledem k povaze diskutované sítě (jednoduchá síť typu LAN) je příklad poměrně zřejmý. Máte-li složitější požadavky a nedaří-li se vám rozchodit program `named`, pak se poohlédněte po knize „DNS and BIND“ od Cricketa Liua a Paula Albitze ([AlbitzLiu92]).

Soubor `named.ca`, který vidíte na obrázku 6.2, ukazuje vzorové záznamy pro kořenové jmenné servery. Typický soubor vyrovnávací paměti obvykle popisuje zhruba deset jmenných serverů. Aktuální seznam jmenných serverů kořenové domény můžete získat pomocí nástroje `nslookup`, který bude popsán na konci této kapitoly.[3]

```
;
; /var/named/named.ca
;                   Nejsme na Internetu a proto nepotřebujeme
;                   kořenové servery. Pro aktivaci těchto záznamů
;                   stačí odstranit středníky.
;
; .                 99999999    IN     NS   NS.NIC.DDN.MIL
; NS.NIC.DDN.MIL    99999999    IN     A    26.3.0.103
; .                 99999999    IN     NS   NS.NASA.GOV
; NS.NASA.GOV       99999999    IN     A    128.102.16.10
```

Obrázek 6.2
Soubor `named.ca`

* Poznámka korektora: Z bezpečnostních důvodů není vhodné používat zdrojové záznamy HINFO.

3 Všimněte si, že vašemu jmennému serveru nemůžete předat dotaz na kořenové servery, dokud nemáte nějaké nainstalovány: Hlava XXII! Abyste z toho vybruslili, můžete buď nařídit nástroji `nslookup`, aby používal odlišný jmenný server, nebo můžete použít jako vstupní bod zdrojový soubor uvedený na obrázku 6.2 a potom získat úplný seznam korektních serverů.

6.2.4 Kontrola nastavení jmenného serveru

Pro kontrolu správné funkce nastavení vašeho jmenného serveru existuje speciální nástroj. Nazývá se `nslookup` a lze ho používat jak interaktivně, tak i z příkazové řádky. V druhém případě ho spustíte následovně:

```
nslookup hostname
```

Nástroj `nslookup` se bude dotazovat jmenného serveru zadaného v souboru `resolv.conf` na název hostitele `hostname`. (Je-li v tomto souboru uveden více než jeden server, pak `nslookup` náhodně zvolí jeden z těchto serverů.)

Interaktivní režim je však mnohem zajímavější. Kromě vyhledávání jednotlivých hostitelů se můžete dotazovat na libovolný typ záznamu systému DNS a dále můžete v rámci příslušné domény přenášet veškeré informace o zónách.

Je-li nástroj `nslookup` vyvolán bez argumentů, zobrazí používaný jmenný server a přepne se do interaktivního režimu. Na příkazové řádce „>“ můžete zadat libovolný název domény, na který by se měl nástroj `nslookup` dotazovat. Implicitně se ptá na záznamy typu A, což jsou ty, které obsahují IP-adresy vztahující se k danému názvu domény.

```
;
; /var/named/named.hosts          Místní hostitelé v pivovaru
;                                 Počátek je vbrew.com
;
@                   IN     SOA    vlager.vbrew.com. (
                                  janet.vbrew.com.
                                  16          ; sériové číslo
                                  86400       ; refresh: denně
                                  3600        ; retry:   1 hodina
                                  3600000     ; expire:  42 dní
                                  604800      ; minimum: 1 týden
                                  )
                    IN     NS     vlager.vbrew.com.
;
; lokální pošta je doručována na vlager
                    IN     MX     10 vlager
;
; zpětnovazebné rozhraní
localhost.          IN     A      127.0.0.1
```

```
; Ethernet pivovaru
vlager              IN     A      191.72.1.1
vlager-if1          IN     CNAME  vlager
; vlager je také news serverem
news                IN     CNAME  vlager
vstout              IN     A      191.72.1.2
vale                IN     A      191.72.1.3
; Ethernet vinárny
vlager-if2          IN     A      191.72.2.1
vbardolino          IN     A      191.72.2.2
vchianti            IN     A      191.72.2.3
vbeaujolais         IN     A      191.72.2.4
```

Obrázek 6.3
Soubor `named.hosts`

```
;
; /var/named/named.local          Reverzní mapování sítě 127.0.0
;                                 Počátek je 0.0.127.in-addr.arpa.
;
@                   IN     SOA    vlager.vbrew.com. (
                                  joe.vbrew.com.
                                  1           ; sériové číslo
                                  360000      ; refresh: 100 hodin
                                  3600        ; retry:   1 hodina
                                  3600000     ; expire:  42 dní
                                  360000      ; minimum: 100 hodin
                                  )
                    IN     NS     vlager.vbrew.com.
1                   IN     PTR    localhost.
```

Obrázek 6.4
Soubor `named.local`

```
;
; /var/named/named.rev            Reverzní mapování našich IP-adres
;                                 Počátek je 72.191.in-addr.arpa.
;
@                   IN     SOA    vlager.vbrew.com. (
```

```
                                joe.vbrew.com.
                                16              ; sériové číslo
                                86400           ; refresh: denně
                                3600            ; retry:   1 hodina
                                3600000         ; expire:  42 dní
                                604800          ; minimum: 1 týden
                                )
                    IN      NS      vlager.vbrew.com.
; pivovar
1.1                 IN      PTR     vlager.vbrew.com.
2.1                 IN      PTR     vstout.vbrew.com.
3.1                 IN      PTR     vale.vbrew.com.
; vinárna
1.2                 IN      PTR     vlager-if1.vbrew.com.
2.2                 IN      PTR     vbardolino.vbrew.com.
3.2                 IN      PTR     vchianti.vbrew.com.
4.2                 IN      PTR     vbeaujolais.vbrew.com.
```

Obrázek 6.5
Soubor `named.rev`

Tento typ je možné změnit příkazem „**`set type`**`=type`“, kde parametr `type` tvoří buď jeden z názvů zdrojových záznamů, které jsme popisovali ve stati 6.2, nebo za něj může být dosazeno klíčové slovo ANY.

S nástrojem `nslookup` můžete vést například následující dialog:

```
$ nslookup
Default Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60

> sunsite.unc.edu
Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60

Non-authoritative answer:
Name:    sunsite.unc.edu
Address:  152.2.22.81
```

Pokusíte-li se dotazovat na název, který nemá přidělenu žádnou IP-adresu, ale v databázi systému DNS byly nalezeny jiné záznamy, vrátí příkaz `nslookup` chybovou zprávu „`No type A records found`“ (nebyly nalezeny žádné záznamy typu A). S pomocí příkazu `nslookup` je možné se dotazovat i na jiné záznamy než typu A. To lze provést příkazem „**`set type`**“. Chcete-li například získat záznam typu SOA na adrese **unc.edu**, měli byste spustit následující sekvenci příkazů:

```
> unc.edu
*** No address (A) records available for unc.edu
Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60

> set type=SOA
> unc.edu
Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60

Non-authoritative answer:
unc.edu
        origin = ns.unc.edu
        mail addr = shava.ns.unc.edu
        serial = 930408
        refresh = 28800 (8 hours)
        retry   = 3600 (1 hour)
        expire  = 1209600 (14 days)
        minimum ttl = 86400 (1 day)
Authoritative answers can be found from:
UNC.EDU nameserver = SAMBA.ACS.UNC.EDU
SAMBA.ACS.UNC.EDU        internet address = 128.109.157.30
```

Podobným způsobem se můžete dotazovat na záznamy typu MX atd. Pokud použijete typ záznamu ANY, vrátí příkaz `nslookup` všechny zdrojové záznamy, které jsou sdruženy s daným názvem.

```
> set type=MX
> unc.edu
Non-authoritative answer:
unc.edu preference = 10, mail exchanger = lambada.oit.unc.edu
lambada.oit.unc.edu      internet address = 152.2.22.80
```

```
Authoritative answers can be found from:
UNC.EDU nameserver = SAMBA.ACS.UNC.EDU
SAMBA.ACS.UNC.EDU        internet address = 128.109.157.30
```

Praktickou aplikací příkazu `nslookup` je kromě ladění jmenného serveru i získávání aktuálního seznamu kořenových jmenných serverů pro soubor `named.ca`. Lze to provést tak, že se budete dotazovat na všechny typy záznamů typu NS, které jsou spojeny s kořenovou doménou:

```
> set type=NS
> .
Name Server:  fb0430.mathematik.th-darmstadt.de
Address:  130.83.2.30

Non-authoritative answer:
(root)  nameserver = NS.INTERNIC.NET
(root)  nameserver = AOS.ARL.ARMY.MIL
(root)  nameserver = C.NYSER.NET
(root)  nameserver = TERP.UMD.EDU
(root)  nameserver = NS.NASA.GOV
(root)  nameserver = NIC.NORDU.NET
(root)  nameserver = NS.NIC.DDN.MIL

Authoritative answers can be found from:
(root)  nameserver = NS.INTERNIC.NET
(root)  nameserver = AOS.ARL.ARMY.MIL
(root)  nameserver = C.NYSER.NET
(root)  nameserver = TERP.UMD.EDU
(root)  nameserver = NS.NASA.GOV
(root)  nameserver = NIC.NORDU.NET
(root)  nameserver = NS.NIC.DDN.MIL

NS.INTERNIC.NET internet address = 198.41.0.4
AOS.ARL.ARMY.MIL         internet address = 128.63.4.82
AOS.ARL.ARMY.MIL         internet address = 192.5.25.82
AOS.ARL.ARMY.MIL         internet address = 26.3.0.29
C.NYSER.NET      internet address = 192.33.4.12
TERP.UMD.EDU     internet address = 128.8.10.90
NS.NASA.GOV      internet address = 128.102.16.10
```

```
NS.NASA.GOV      internet address = 192.52.195.10
NS.NASA.GOV      internet address = 45.13.10.121
NIC.NORDU.NET    internet address = 192.36.148.17
NS.NIC.DDN.MIL   internet address = 192.112.36.4
```

Kompletní seznam příkazů, které lze použít ve spojitosti s nástrojem `nslookup`, získáte příkazem `help`, který musí být zadán během interaktivní práce s nástrojem `nslookup`.

6.2.5 Další užitečné nástroje

Existuje několik nástrojů, které vám mohou pomoci při úkolech, s nimiž se setkáte jako správce služby BIND. Zde si popíšeme dva z nich. Chcete-li získat informace o jejich použití, nahlédněte do dokumentace, která je dodávána spolu s nimi.

Nástroj `hostcvt` pomáhá při prvotní konfiguraci služby BIND. Provádí konverzi souboru `/etc/hosts` do hlavních souborů pro program `named`. Vygeneruje data jak pro přímé mapování (typ záznamu A), tak i pro zpětné mapování (typ záznamu PTR) a postará se i o přezdívky. Samozřejmě, že nemůže udělat vše. Stále se budete muset postarat například o hodnoty časových intervalů v záznamu typu SOA, o přidání záznamů typu MX atp. Přesto vám však může ušetřit několik aspirinů. Nástroj `hostcvt` je částí zdrojového kódu služby BIND, ale je možné ho nalézt i jako samostatný balík na některých linuxových FTP serverech.

Jakmile nakonfigurujete vlastní jmenný server, budete si ho zřejmě chtít otestovat. Ideálním (a podle mých vědomostí) zřejmě i jediným nástrojem, který byl vytvořen za tímto účelem, je `dnswalk`. Jedná se o balík napsaný v jazyce Perl, který projde vaši databázi systému DNS, vyhledá běžné chyby a ověří konzistenci informací. Nástroj `dnswalk` byl teprve nedávno uvolněn skupině **comp.sources.misc** a měl by být dostupný na všech serverech, které tento server zrcadlí (neznáte-li ve svém okolí žádný takový systém, pak zmiňovaný balík bezpečně najdete na serveru **ftp.uu.net**).

Kapitola

7

IP po sériové lince

Protokoly pro sériové linky, SLIP a PPP, umožňují připojení k Internetu i chudším vrstvám. Kromě modemu a sériového portu vybaveného vyrovnávací pamětí typu FIFO již není zapotřebí žádný další hardware. Jeho použití není o nic složitější, než používání poštovní schránky, a přitom připojení pomocí modemu nabízí za rozumnou cenu stále více soukromých firem.

V systému Linux je dostupný jak ovladač pro protokol SLIP, tak i ovladač pro protokol PPP. Protokol SLIP je v něm zabudován už poměrně dlouhou dobu a tudíž pracuje poměrně spolehlivě. Ovladač PPP vytvořili teprve nedávno Michael Callahan a Al Longyear. Tento ovladač bude popsán v další kapitole.

7.1 Obecné požadavky

Abyste mohli používat protokoly SLIP nebo PPP, bude třeba nastavit několik základních síťových vlastností, které byly popsány v předchozích kapitolách. Přinejmenším musíte nastavit zpětnovazebné rozhraní a povolit službu pro rozlišení názvů. Když se budete připojovat k Internetu, budete samozřejmě chtít používat systém DNS. Nejjednodušší způsob, jak toho docílit, spočívá ve vložení adresy nějakého jmenného serveru do souboru `resolv.conf`. Jakmile bude aktivováno spojení pomocí protokolu SLIP, bude poslán dotaz tomuto serveru. Čím blíže bude název tohoto serveru od místa, ze kterého voláte, tím lépe.

Toto řešení však není optimální, protože všechna vyhledávání názvů budou využívat spojení SLIP/PPP. Pokud vám dělá starosti šířka pásma, která je k tomuto účelu využívána, můžete nastavit tzv. `caching-only` jmenný server. Ten ve skutečnosti neobsluhuje doménu, ale působí pouze jako prostředník pro všechny dotazy systému DNS, které pochází z vašeho hos-

titele. Výhodou tohoto schématu je vytvoření vyrovnávací paměti, kdy je většina dotazů posílána po sériové lince pouze jednou. Soubor `named.boot` pro tento server vypadá podobně jako následující výpis:

```
; soubor named.boot pro caching-only server
directory                          /var/named
primary      0.0.127.in-addr.arpa  db.127.0.0 ; zpětnovazebná síť
cache         .                    db.cache   ; kořenové servery
```

K tomuto souboru `named.boot` musíte v souboru `db.cache` nastavit korektní sezname kořenových jmenných serverů. To je popsáno na konci kapitoly Konfigurace resolveru.

7.2 Provozování protokolu SLIP

Dial-up IP-servery často nabízejí službu SLIP pomocí speciálních uživatelských účtů. Po přihlášení k takovému účtu nejste vpuštěni do nějakého obecného uživatelského rozhraní; namísto toho je spuštěn program nebo skript rozhraní, který povolí na serveru pro danou sériovou linku ovladač SLIP a nakonfiguruje patřičné síťové rozhraní. Totéž se musí provést na vaší straně spojení.

V některých operačních systémech je ovladač SLIP speciálním uživatelským programem; v operačním systému Linux je součástí jádra systému, takže je výrazně rychlejší. Je však nutné, aby byla sériová linka explicitně převedena do režimu SLIP. To se provede pomocí speciálního *tty* režimu linky, tzv. SLIPDISC. Je-li zařízení *tty* v normálním režimu linky (DISC0), bude si vyměňovat data pouze s uživatelskými procesy pomocí standardních volání *read(2)* a *write(2)* a ovladač SLIP nebude schopen zapisovat nebo číst ze zařízení *tty*. V režimu SLIPDISC jsou role obráceny: nyní nebude moci zapisovat nebo číst ze zařízení žádný uživatelský proces, ale všechna data přicházející na sériový port budou přímo předána ovladači SLIP.

Vlastní ovladač protokolu SLIP rozumí množství variací protokolu SLIP. Kromě běžného protokolu SLIP ovládá také protokol CSLIP, který u odcházejících IP-paketů provádí tzv. Van Jacobsonovu kompresi hlaviček.[1] To výrazně zvyšuje propustnost dat při interaktivní práci. Mimoto existují i šestibitové verze obou těchto protokolů.

Jednoduchý způsob, jak zkonvertovat sériovou linku do režimu SLIP, spočívá ve využití nástroje `slattach`. Předpokládejme, že máte modem nastaven na `/dev/cua3` a že jste se úspěšně přihlásili k serveru SLIP. Potom spustíte následující příkaz:

[1] Van Jacobsonova komprese hlavičky je popsána v RFC 1441.

```
# slattach /dev/cua3 &
```

Tento příkaz přepne linku `cua3` do režimu SLIPDISC a připojí ji k jednomu ze síťových rozhraní SLIP. Je-li to vaše první aktivní spojení pomocí protokolu SLIP, bude linka připojena k rozhraní *sl0*; další bude připojena k rozhraní *sl1* atd. Aktuální verze jader operačního systému podporují až osm současných spojení pomocí protokolu SLIP.

Implicitní zapouzdření, které vybere příkaz `slattach` je protokol CSLIP. K volbě některého jiného režimu lze použít argument `-p`. Chcete-li používat standardní protokol SLIP (bez komprese), pak byste měli použít následující příkaz:

```
# slattach -p slip /dev/cua3 &
```

Další volitelné režimy jsou: `cslip`, `slip6`, `cslip6` (pro šestibitové verze protokolu SLIP) a `adaptive` pro adaptivní protokol SLIP. Poslední volba nechává nalezení typu zapouzdření protokolu SLIP, který používá vzdálený počítač, na jádru operačního systému.

Pamatujte, že musíte používat stejný typ zapouzdření, jaký používá váš protějšek. Pokud například hostitel **cowslip** používá protokol CSLIP, musíte ho použít také. Neshody jednotlivých verzí protokolu SLIP mohou například způsobit, že příkaz `ping` použitý na vzdáleného hostitele neobdrží zpátky žádné pakety. Pokud nějaký jiný hostitel spustí příkaz `ping` s vaším jménem, může se na vaší konzole objevit zpráva „`Can't build ICMP header`." Jeden ze způsobů, jak se vyhnout takovýmto komplikacím, spočívá v použití adaptivního protokolu SLIP.

Příkaz `slattach` ale nepovoluje jen použití protokolu SLIP, ale také jiné druhy protokolů, jako jsou například protokoly PPP nebo KISS (další protokol používaný v síti ham radio). Chcete-li více detailů, nahlédněte prosím na manuálovou stránku k příkazu `slattach(8)`.

Jakmile lince přiřadíte ovladač protokolu SLIP, musíte nakonfigurovat síťové rozhraní. Opět k tomu využijeme standardní příkazy `ifconfig` a `route`. Předpokládejme, že jsme se z brány **vlager** připojili k serveru jménem **cowslip**. Pak je nutné spustit následující sekvenci příkazů:

```
# ifconfig sl0 vlager pointopoint cowslip
# route add cowslip
# route add default gw cowslip
```

První příkaz nastaví rozhraní na spojení s hostitelem **cowslip** na typ point-to-point, druhý a třetí příkaz přidá směrování na hostitele **cowslip** a implicitní směr, přičemž bude hostitele **cowslip** využívat jako bránu.

Při rušení spojení pomocí protokolu SLIP musíte nejprve odstranit všechna směrování přes hostitele **cowslip**. K tomu slouží příkaz `route` s volbou `del`. Potom je nutné odpojit rozhraní a poslat nástroji `slattach` signál zavěšení. Následně byste měli znovu zavěsit modem pomocí svého terminálového programu:

```
# route del default
# route del cowslip
# ifconfig sl0 down
# kill -HUP 516
```

7.3 Použití nástroje dip

Až sem to bylo poměrně jednoduché. Přesto však možná budete chtít výše uvedené kroky zautomatizovat natolik, aby bylo možné celý postup vyvolat pouze jediným příkazem. K tomuto účelu slouží nástroj `dip`.[2] Aktuální číslo verze tohoto nástroje je 3.3.7. Ta však byla velmi často upravována a pozměňována, takže vlastně nemůže být o nějakém nástroji `dip` ani řeč. Doufejme, že tyto odlišnosti budou začleněny do budoucí verze.

Nástroj `dip` je interpretem jednoduchého skriptového jazyka, který se může starat o modem, nastavovat linku do režimu SLIP a konfigurovat různá rozhraní. Je to poměrně primitivní a omezující, ale ve většině případů to stačí. Doufejme, že některá nová verze nástroje `dip` bude obsahovat všestrannější jazyk.

Aby bylo možné konfigurovat rozhraní SLIP, musí být nástroji `dip` přidělena práva **superuživatele**. Teď to možná vypadá, že pokud nastavíte nástroji `dip` práva **superuživatele**, budou se moci všichni uživatelé spojit s libovolným serverem SLIP, aniž by měli přidělena přístupová práva **superuživatele**. To je velmi nebezpečné, protože nastavení falešných rozhraní a implicitních směrů pomocí nástroje `dip` může výrazně poškodit směrování ve vaší síti. A co je ještě horší, vaši uživatelé tím získají možnost spojení s libovolným serverem SLIP a budou moci provádět nebezpečné útoky na vaši síť. Pokud tedy chcete umožnit svým uživatelům provozovat spojení pomocí protokolu SLIP, napište pro každý plánovaný server SLIP malé obslužné programy a teprve z těchto programů volejte nástroj `dip` s konkrétním skriptem, který založí spojení SLIP. Až potom lze těmto programům bezpečně nastavit práva **superuživatele**.[3]

2 `dip` znamená *Dialup IP*. Tento nástroj napsal Fred van Kempen.

3 Práva lze nastavit i pro příkaz `diplogin`. Podrobnosti najdete na konci této kapitoly.

7.3.1 Vzorový skript

Na obrázku 7.1 vidíte vzorový skript. Lze ho použít ke spojení s hostitelem **cowslip**. Nástroj `dip` je možné spustit s argumentem, který bude obsahovat název skriptu:

```
# dip cowslip.dip
DIP: Dialup IP Protocol Driver version 3.3.7 (12/13/93)
Written by Fred N. van Kempen, MicroWalt Corporation

connected to cowslip.moo.com with addr 193.174.7.129
#
```

Po spojení s hostitelem *cowslip* a povolení protokolu SLIP se proces `dip` odpojí od terminálu a dále bude spuštěn na pozadí. Potom můžete začít používat normální síťové služby po lince SLIP. Chcete-li spojení ukončit, vyvolejte nástroj *dip* s parametrem `-k`. Tento příkaz pošle procesu `dip` signál zavěšení, přičemž použije záznamy identifikačního čísla procesu `dip`, které se nacházejí v souboru `/etc/dip.pid`:[4]

```
# Jednoduchý dip script pro volání hostitele cowslip
# Nastav místní a vzdálené jméno a adresu
get $local vlager
get $remote cowslip

port cua3                  # volba sériového portu
speed 38400                # maximální rychlost
modem HAYES                # typ modemu
reset                      # reset modemu a tty
flush

# Příprava na vytočení
send ATQ0V1E1X1\r
wait OK 2
if $errlvl != 0 goto error
dial 41988
if $errlvl != 0 goto error
wait CONNECT 60
if $errlvl != 0 goto error
```

4 Další zábavu s třípísmennými akronymy přináší konference **alt.tla**.

```
# Nyní jsme spojeni
sleep 3
send \r\n\r\n
wait ogin: 10
if $errlvl != 0 goto error
send Svlager\r
wait ssword: 5
if $errlvl != 0 goto error
send hey-jude\n
wait running 30
if $errlvl != 0 goto error

# Nyní jsme přihlášeni a vzdálený hostitel spustil SLIP.
print Connected to $remote with address $rmtip
default                    # Nastavení implicitní směrové cesty
mode CSLIP                 # Spuštění protokolu SLIP
# v případě chyby pokračujte zde

error:
print SLIP to $remote failed.
```

Obrázek 7.1
Vzorový skript k nástroji `dip`


```
# kill -k
```

Ve skriptovém jazyku nástroje `dip` znamenají klíčová slova, před nimiž je uveden symbol dolaru, názvy proměnných. Nástroj `dip` má předdefinovanou skupinu proměnných, která bude uvedena níže. Například proměnné *$remote* a *$local* obsahují názvy místního a vzdáleného hostitele, kteří se účastní spojení pomocí protokolu SLIP.

První dva příkazy ve vzorovém skriptu jsou příkazy `get`, které reprezentují způsob, jakým nástroj `dip` nastavuje proměnné. Zde je nastaven název místního a vzdáleného hostitele **vlager**, resp. **cowslip**.

Dalších pět řádek nastavuje terminálovou linku a modem. Příkaz `reset` pošle modemu inicializační řetězec; u modemů kompatibilních se standardem Hayes odpovídá tento řetězec příkazu *ATZ*. Další příkaz nastaví odpovědi modemu tak, aby přihlašovací sekvence, která následuje na dalších řádcích, pracovala správně. Tato přihlašovací sekvence je poměrně přehledná: nejdříve se vytočí číslo 41 988, což je telefonní číslo hostitele *cowslip* a hostitel se přihlásí na

účet *Svlager* s heslem *hey-jude*. Příkaz `wait` způsobí prodlevu nástroje `dip`, protože tento bude čekat na vstup odpovídající řetězci, který je uveden v prvním argumentu; číslo uvedené v druhém argumentu odpovídá době, po kterou se bude čekat v případě, že se nepodaří získat hodnotu odpovídající prvnímu argumentu. Příkaz `if` průběžně uváděný v přihlašovací proceduře kontroluje, zda při spouštění příkazu nedošlo k chybě.

Posledními příkazy spouštěnými po přihlášení jsou `default`, jenž nastaví u spojení SLIP implicitní směr na všechny hostitele, a `mode`, jenž povolí na dané lince režim SLIP a nakonfiguruje rozhraní a směrovací tabulku.

7.3.2 Manuál k nástroji dip

I když je nástroj `dip` velmi rozšířen, není zatím příliš dobře zdokumentován. Proto v této stati uvádíme popis většiny příkazů nástroje `dip`. Přehled všech dostupných příkazů získáte, když nástroj `dip` spustíte v testovacím režimu zadáním příkazu `help`. Chcete-li zjistit syntaxi příkazu, stačí ho zadat bez argumentů; samozřejmě, že výše uvedené nefunguje u příkazů, které nemají žádné argumenty.

```
$dip -t
DIP: Dialup IP Protocol Driver version 3.3.7 (12/13/93)
Written by Fred N. van Kempen, MicroWalt Corporation

DIP> help
DIP knows about the following commands:

databits     default      dial         echo         flush
get          goto         help         if           init
mode         modem        parity       print        port
reset        send         sleep        speed        stopbits
term         wait

DIP> echo
Usage: echo on|off
DIP>
```

V průběhu následujícího výkladu budou řádky začínající řetězcem `DIP>` ukazovat, jak se zadává příslušný příkaz v testovacím režimu a následující řádky pak budou zobrazovat jeho výstup. Příklady, které příkazovou řádku neobsahují, by měly být chápány jako výpisy skriptu.

Příkazy pro modem

Nástroj `dip` poskytuje značný počet příkazů pro konfiguraci sériové linky a modemu. Význam některých z nich je zřejmý, například příkaz `port`, který vybírá sériový port, nebo příkazy `speed, databits, stopbits a parity`, které nastavují obecné parametry linky.

Příkaz `modem` volí typ modemu. V současné době je jediným podporovaným typem modemu typ *HAYES* (velká písmena jsou vyžadována). Nástroji `dip` musíte poskytnout typ modemu, v opačném případě by totiž odmítl spustit příkazy `dial` a `reset`. Příkaz `reset` posílá modemu vstupní inicializační řetězec; použitý typ řetězce závisí na vybraném typu modemu. U modemů kompatibilních se standardem HAYES je tímto řetězcem příkaz *ATZ*.

Příkaz `flush` slouží ke smazání všech odpovědí, které modem v minulosti poslal. V opačném případě by mohl být přihlašovací skript, který následuje po příkazu `reset`, špatný, protože modem by mohl číst odpovědi OK z předchozích příkazů.

Příkaz `init` vybírá inicializační řetězec, který bude předán modemu před začátkem vytáčení. Implicitním řetězcem pro modemy kompatibilní se standardem HAYES je „*ATE0 Q0 V1 X1*“, který zapne zobrazování příkazů, dlouhé kódy výsledků a nastaví volání naslepo (bez detekce vytáčecího tónu).

Nakonec pošle příkaz `dial` modemu inicializační řetězec a vytočí číslo vzdáleného systému. Implicitním vytáčecím příkazem pro modemy kompatibilní se standardem HAYES je příkaz *ATD*.

Příkazy echo a term

Příkaz `echo` slouží jako ladící prostředek, protože použití příkazu `echo on` způsobí, že nástroj `dip` bude na konzole zobrazovat vše, co je posíláno na sériové zařízení. Tuto vlastnost lze opět zrušit příkazem `echo off`.

Nástroj `dip` také umožňuje dočasně opustit skriptový režim a vstoupit do terminálového režimu. V tomto režimu lze nástroj `dip` používat jako jakýkoliv jiný běžný terminálový program. Můžete v něm zapisovat na sériovou linku nebo z ní číst. Chcete-li tento režim opustit, stiskněte kombinaci kláves **Ctrl +]**.

Příkaz get

Příkaz `get` používá nástroj `dip` k nastavování proměnných. Nejjednodušším způsobem je nastavit proměnnou jako konstantu, což je používáno ve výše uvedeném příkladu. Můžete ale také požádat uživatele, aby hodnotu zadal. K tomu slouží klíčové slovo `ask`, které je nutno uvést na místě hodnoty proměnné:

```
DIP> get $local ask
Enter the value for $local: _
```

Třetí metoda spočívá v pokusu o získání této hodnoty ze vzdáleného hostitele. Na první pohled to vypadá zvláštně, ale v některých případech je tento způsob velmi užitečný: Některé servery SLIP vám nedovolí při spojení pomocí protokolu SLIP používat vlastní IP-adresu. Místo toho vám při každém přihlášení přidělí nějakou adresu ze seznamu adres a zobrazí zprávu, která vás bude o přidělené adrese informovat. Pokud tato zpráva vypadá podobně jako „`Your address:  193.174.7.202`“, umožní vám následující část kódu nástroje `dip` zvolit vlastní adresu:

```
... login chat ....
wait address: 10
get $locip remote
```

Příkaz print

Tento příkaz umožňuje vypsat text na konzolu, ze které byl spuštěn nástroj `dip`. V rámci příkazu `print` lze použít libovolné proměnné definované v nástroji `dip`, například:

```
DIP> print Using port $port at speed $speed
Using port cua3 at speed 38400
```

Názvy proměnných

Nástroj `dip` rozumí pouze předdefinované skupině proměnných. Název proměnné vždy začíná symbolem dolaru a musí být psán malými písmeny.

Proměnné *$local* a *$locip* obsahují název místního hostitele a IP-adresu. Po nastavení názvu hostitele uloží nástroj `dip` do proměnné *$local* kanonický název hostitele a zároveň přiřadí proměnné *$locip* odpovídající IP-adresu. Analogický proces probíhá, když nastavujete proměnnou *$locip*.

Proměnné *$remote* a *$rmtip* provádí totéž s názvem vzdáleného hostitele a jeho IP-adresou. Proměnná *$mtu* obsahuje hodnotu MTU pro dané spojení.

Tyto proměnné jsou jedinými proměnnými, kterým mohou být přímo přidělovány hodnoty pomocí příkazu `get`. Dalších proměnné lze nastavovat pouze za pomoci odpovídajících příkazů, nicméně i tyto proměnné lze používat v rámci příkazu `print`; konrétně se jedná o proměnné *$modem*, *$port* a *$speed*.

Proměnná *$errlvl* umožňuje přistupovat k výsledku naposledy spuštěného příkazu. Návratová hodnota rovná nula značí úspěšnou operaci, zatímco nenulová hodnota naznačuje chybnou operaci.

Příkazy if a goto

Příkaz `if` představuje podmíněné větvení. Jeho syntaxe je následující:

```
if var op number goto label
```

Výraz musí být jednoduchým porovnáním jedné z proměnných *$errlvl*, *$locip* a *$rmtip* a celočíselné hodnoty; operátorem `op` může být jeden z následujících operátorů: `==`, `!=`, `<`, `>`, `<=` a `>=`.

Příkaz `goto` umožní, aby provádění skriptu pokračovalo na řádce následující po daném návěští. Návěští musí být uvedeno na začátku řádku a bezprostředně za ním musí následovat dvojtečka.

Příkazy send, wait a sleep

Tyto příkazy pomáhají implementovat do nástroje `dip` jednoduché skripty s rozhovorem. Příkaz `send` zapíše své argumenty na sériovou linku. Nepodporuje žádné proměnné, avšak rozumí všem kombinacím znaků s převráceným lomítkem, které pocházejí z jazyka C, jako je například *\n* a *\b*. Znak vlnovky (~) je používán jako zkratka pro návrat vozíku/nová řádka (`CR/LF`).

Argumentem příkazu `wait` je slovo a jeho funkce spočívá v monitorování všech vstupů po sériové lince, dokud se dané slovo neobjeví. Vlastní slovo nemůže obsahovat žádné mezery. K příkazu `wait` můžete přidat i druhý nepovinný argument, který bude reprezentovat délku trvání příkazu; pokud nebude očekávané slovo obdrženo v daném časovém limitu, příkaz skončí a nastaví proměnnou *$errlvl* na hodnotu 1.

Příkaz `sleep` slouží k nastavení časové prodlevy, například aby nástroj `dip` trpělivě vyčkal na dokončení přihlašovací procedury. Tento interval je opět zadáván v sekundách.

Příkazy mode a default

Tyto příkazy se používají k přepnutí sériové linky do režimu SLIP a ke konfiguraci rozhraní. Příkaz `mode` je posledním příkazem spuštěným v nástroji `dip` předtím, než se nástroj `dip` přepne do režimu démona.

Příkaz `mode` přijímá jako argument název protokolu. Nástroj `dip` v současné době akceptuje jako korektní názvy protokolů *SLIP* a *CSLIP*. Aktuální verze nástroje `dip` bohužel neakceptuje adaptivní protokol SLIP.

Jakmile na sériové lince povolíte režim SLIP, spustí nástroj `dip` příkaz `ifconfig`, aby nastavil rozhraní jako spojení typu point-to-point, a dále zavolá příkaz `route`, který nastaví směrování ke vzdálenému hostiteli.

Pokud skript navíc spustí před příkazem `mode` i příkaz `default`, nastaví nástroj `dip` také implicitní směr pro spojení pomocí protokolu SLIP.

7.4 Spouštění v režimu server

Nastavení klienta s protokolem SLIP bylo poměrně obtížné. Nastavení protějšku, konkrétně takové konfigurace vašeho hostitele, aby se choval jako server SLIP, je mnohem jednodušší.

Jedním ze způsobů, jak to provést, je použit nástroj `dip` v režimu server. Toho dosáhnete za pomoci nástroje `diplogin`. Jeho hlavním konfiguračním souborem je soubor `/etc/diphosts`, který sdružuje přihlašovací jména s adresou, která je danému hostiteli přidělena. Alternativně můžete používat i nástroj `sliplogin`, který je odvozen z balíku BSD a umožňuje mnohem pružnější konfigurační schéma, na jehož základě lze spouštět skripty daného rozhraní kdykoliv se hostitel připojí nebo odpojí. V současné době je tento nástroj ve fázi beta.

Oba programy vyžadují, abyste pro každého klienta SLIP nastavili jeden přihlašovací účet. Předpokládejme, že poskytujete službu SLIP panu Arthuru Dentovi, který má adresu **dent.beta.com**. Účet s názvem **dent** vytvoříte tak, že do souboru `passwd` přidáte následující řádku:

```
dent:*:501:60:Arthur Dent's SLIP account:/tmp:/usr/sbin/diplogin
```

Potom byste měli pomocí utility `passwd` nastavit heslo pro účet **dent**.

Když se nyní uživatel **dent** přihlásí, spustí se nástroj `dip` v režimu server. Aby nástroj `dip` zjistil, zda je tento uživatel oprávněn používat službu SLIP, vyhledá jeho jméno v souboru `/etc/diphosts`. Tento soubor popisuje přístupová práva a parametry spojení každého uživatele služby SLIP. Vzorová položka pro uživatele **dent** vypadá takto:

```
dent::dent.beta.com:Arthur Dent:SLIP,296
```

První pole oddělené dvojtečkou reprezentuje název účtu, pod kterým se musí uživatel přihlásit. Druhé pole může obsahovat dodatečné heslo (viz níže). Třetí pole obsahuje název hostitele nebo IP-adresu volajícího hostitele. Dále následuje informační pole, které nemá žádný speciální význam (prozatím). Poslední pole popisuje parametry spojení. Obsahuje seznam oddělený čárkou, který určuje používaný protokol (momentálně buď *SLIP,* nebo *CSLIP*), a za ním následuje hodnota MTU.

Když se uživatel **dent** přihlásí, nástroj `diplogin` si o něm vytáhne informace ze souboru `diphosts`, a pokud není druhé pole prázdné, vyzve ho k zadání „externího bezpečnostního hesla“. Řetězec zadaný uživatelem pak porovná s (nezašifrovaným) heslem v souboru `diphosts`. Pokud nesouhlasí, bude pokus o přihlášení zamítnut.

V opačném případě začne nástroj `diplogin` s přepnutím sériové linky do režimu CSLIP nebo SLIP a nastaví také rozhraní a směrování. Toto spojení trvá až do té doby, kdy se uživatel odpojí a modem zavěsí linku. Poté nástroj `diplogin` přepne linku zpět do normálního režimu a skončí.

Nástroj `diplogin` vyžaduje speciální uživatelská práva. Pokud nástroji `dip` nepřidělíte práva **superuživatel**, měli byste nástroje `diplogin` a `dip` od sebe oddělit (například vytvořením samostatné kopie nástroje `diplogin`) a nepoužívat je při jednoduchém spojení společně. V takovém případě mohou být nástroji `diplogin` přidělena přístupová práva, která neovlivní práva nástroje `dip`.

Kapitola

8

Point-to-Point Protokol

8.1 Vysvětlení protokolu PPP

Protokol PPP slouží, stejně jako protokol SLIP, k posílání datagramů po sériové lince, avšak odstraňuje spoustu nedostatků, kterými trpí protokol SLIP. Umožňuje, aby si komunikující strany na počátku spojení dohodly parametry spojení, kterými mohou být například IP-adresa nebo maximální velikost datagramu. Dále poskytuje protokol PPP možnost ověření totožnosti klienta. Pro každou z těchto vlastností má protokol PPP samostatný protokol. V této kapitole si tyto základní stavební prvky protokolu PPP popíšeme. Kapitola však zdaleka neposkytuje veškeré informace; chcete-li se toho dozvědět o protokolu PPP více, měli byste si přečíst jeho specifikaci v RFC 1548 a dále přibližně tucet doprovodných RFC.[1]

Na nejnižší hladině protokolu PPP se nachází tzv. *Vysokoúrovňový protokol pro řízení datového spojení* (*High-Level Data Link Control Protocol*), zkráceně HDLC[2], který definuje pole jednotlivých rámců protokolu PPP a poskytuje 16bitový kontrolní součet. Na rozdíl od primitivnějšího zapouzdření v protokolu SLIP může rámec v protokolu PPP obsahovat pakety i jiných protokolů, než IP, například protokolu IPX sítě Novell nebo protokolu Appletalk. Této vlastnosti je v protokolu PPP dosaženo tak, že k základnímu rámci protokolu HDLC je přidáno speciální protokolové pole, které identifikuje typ paketu přenášeného daným rámcem.

Protokol LCP, neboli protokol pro řízení spojení (Link Control Protocol), se používá nad protokolem HDLC a používá se ke sjednávání parametrů týkajících se datového spojení, jako je například maximální příjmová jednotka (Maximum Receive Unit – MRU), jež uvádí maximální velikost datagramu, kterou je jedna ze stran ochotná přijímat.

---

1 Významné RFC jsou uvedeny na konci této knihy v bibliografii s poznámkami.

2 Ve skutečnosti je protokol HDLC mnohem obecnější protokol, který navrhla organizace International Standards Organization (ISO).

Důležitým krokem ve fázi konfigurace spojení pomocí protokolu PPP je ověření totožnosti klienta. Ačkoliv není povinné, je u většiny linek nabízejících připojení pomocí modemu de facto nezbytné. Volaný hostitel (server) obvykle vyzve klienta k ověření své totožnosti za pomocí nějakého tajného klíče. Není-li volající schopen poskytnout správný tajný klíč, bude spojení ukončeno. U protokolu PPP funguje ověřování totožnosti oběma směry; to znamená, že i klient může požádat server, aby prokázal svou totožnost. Obě tyto ověřovací procedury jsou na sobě vzájemně nezávislé. Pro účely těchto dvou různých způsobů ověřování jsou k dispozici dva protokoly, o kterých se ještě zmíníme. Nazývají se protokol pro ověření hesla (Password Authentication Protocol), zkráceně PAP, a protokol na ověření inicializační výzvy (Challenge Handshake Authentication Protocol – CHAP).

Každý síťový protokol, který je směrován po datové lince, například protokoly IP, AppleTalk atd., je konfigurován dynamicky za pomoci odpovídajícího protokolu pro řízení sítě (Network Control Protocol – NCP). Například při posílání IP-datagramu po lince si musí nejprve oba hostitelé s protokolem PPP dohodnout, jakou IP-adresu bude každý z nich používat. Řídicím protokolem, který se k tomuto účelu používá, je protokol IPCP, neboli protokol na řízení internetového protokolu (Internet Protocol Control Protocol).

Kromě vlastního posílání IP-datagramů po lince podporuje protokol PPP také Van Jacobsonovu kompresi hlaviček IP-datagramů. To je technika používaná ke zmenšování velikosti hlaviček TCP-paketů až na tři bajty. Tato technika se používá také v protokolu CSLIP a obecně je známá pod názvem VJ komprese hlaviček. I použití komprese může být sjednáno při spuštění pomocí protokolu IPCP.

8.2 Protokol PPP v Linuxu

V Linuxu je funkce protokolu rozdělena na dvě části, na vysokoúrovňový ovladač HDLC, který se nachází v jádru operačního systému, a na démona uživatelského prostoru `pppd`, který se stará o různé řídicí protokoly.

Ovladač protokolu PPP v jádru operačního systému napsal Michael Callahan. Démon `pppd` byl odvozen z volně šiřitelné implementace protokolu PPP v počítačích Sun a 386BSD, jehož autorem je Drew Perkins a kol. a nyní jej spravuje Paul Mackerras. Na platformu Linuxu ho převedl Al Longyear.[3] Program `chat` napsal Karl Fox.[4]

[3] Oba autoři říkají, že jsou čas od času velmi zaneprázdnění. Máte-li nějaké obecné dotazy týkající se protokolu PPP, uděláte nejlépe, když se na ně zeptáte lidí v kanálu NET, kteří jsou přihlášeni v poštovní konferenci fandů Linuxu.

[4] **karl@morningstar.com**.

Protokol PPP je, stejně jako protokol SLIP, implementován pomocí speciálního režimu linky. Chcete-li používat nějakou sériovou linku jako linku s protokolem PPP, musíte nejprve obvyklým způsobem vytvořit spojení pomocí modemu a následně převést linku do režimu protokolu PPP. V tomto režimu budou všechna příchozí data podstoupena ovladači protokolu PPP, který ověří platnost rámců protokolu HDLC (každý rámec HDLC je doplněn 16bitovým kontrolním součtem), rozbalí je a odešle. V současné době je schopen spravovat IP-datagramy, volitelně i použití Van Jacobsonovy komprese hlaviček. Jakmile bude Linux podporovat i protokol IPX, dojde i k rozšíření ovladače PPP o správu IPX paketů.* Ovladači jádra operačního systému pomáhá démon `pppd`, což je démon protokolu PPP, který provádí veškerou inicializační fázi a fázi ověřování totožnosti, což je nutné před zahájením provozu po příslušné lince. Chování démona `pppd` je možné doladit pomocí mnoha parametrů. Jelikož je ovladač protokolu PPP poměrně složitý, není možné popsat všechny tyto parametry v jediné kapitole. Tato kniha tak nepokrývá všechny aspekty démona `pppd`, poskytuje jen stručný úvod. Chcete-li se o tomto problému dozvědět více, nahlédněte do manuálu a do souborů `README`, které jsou součástí distribuce zdrojového kódu démona `pppd`, a tam byste měli nalézt většinu dotazů, jež nejsou součástí této kapitoly. Pokud budete mít problémy i po přečtení těchto materiálů, obraťte se na konferenci **comp.protocols.ppp**, která sdružuje většinu lidí zainteresovaných na vývoji démona `pppd`.

8.3 Spuštění démona pppd

Když se chcete připojit na Internet pomocí protokolu PPP, musíte nejdříve nastavit základní síťovou podporu, jako je zpětnovazebné zařízení a resolver. Oba druhy síťové podpory byly probírány v předchozích kapitolách. S použitím systému DNS souvisí ještě některé věci, o kterých je třeba se zmínit; najdete je v kapitole věnované protokolu SLIP.

V úvodním příkladu, který se bude zabývat navázáním spojení PPP pomocí démona `pppd`, předpokládáme, že jste opět na hostiteli **vlager**. Již jste vytočili server PPP s názvem **c3po** a přihlásili jste se na účet **ppp**. Server **c3po** již aktivoval svůj ovladač protokolu PPP. Po ukončení komunikačního programu, který jste použili k vytočení čísla serveru, spustíte následující příkaz:

```
# pppd /dev/cua3 38400 crtscts defaultroute
```

Tento příkaz přepne sériovou linku `cua3` do režimu PPP a naváže IP-spojení se serverem **c3po**. Přenosová rychlost použitá sériovým portem bude 38 400 bps. Parametr `crtscts` zapíná na daném portu hardwarové řízení toku dat, které u rychlostí nad 9 600 bps musí být rozhodně zapnuto.

* Poznámka korektora: Linux v současné době protokol IPX plně podporuje.

Démon pppd si ihned po spuštění domluví některé charakteristiky se svým protějšekem. K tomu použije protokol LCP. Při sjednávání charakteristik budou obvykle fungovat implicitní parametry, takže se jimi zde nemusíme dále zabývat. V další kapitole se na protokol LCP podíváme detailněji.

Budeme také předpokládat, že server **c3po** od nás nebude vyžadovat žádný typ ověřování totožnosti, čímž máme konfigurační fázi úspěšně za sebou.

Následně domluví démon pppd se svým protějškem parametry IP, k čemuž použije protokol IPCP, což je protokol řídící IP. Protože jsme démonu pppd nepředali žádnou konkrétní IP-adresu, pokusí se ji získat tak, že nechá resolver vyhledat místní název hostitele. Potom si oba hostitelé sdělí své IP-adresy.

Tato implicitní nastavení jsou zpravidla dostačující. Dokonce i v případě, že je váš počítač připojen do sítě Ethernet, můžete použít stejnou IP-adresu jak pro Ethernet, tak i pro rozhraní PPP. Přesto vám démon pppd dovolí použít odlišnou adresu, případně může požádat váš protějšek, aby použil nějakou konkrétní adresu. Tyto volby jsou rozebírány dále.

Po úspěšném dokončení fáze nastavení protokolu IPCP připraví démon pppd síťovou vrstvu, která se bude používat při spojení pomocí protokolu PPP. Nejprve nastaví síťové rozhraní PPP jako spojení typu point-to-point, a to tak, že pro první aktivní spojení pomocí protokolu PPP použije rozhraní ppp0, pro druhé aktivní spojení ppp1 atd. Dále nastaví položku ve směrovací tabulce, která bude odkazovat na hostitele na druhém konci spojení. V dříve zmíněném příkladu nastaví démon pppd implicitní směr na server **c3po**, protože mu byla přiřazena volba defaultroute.[5] Tato volba způsobí, že všechny datagramy poslané hostitelům, kteří se nenacházejí ve vaší místní síti, budou poslány na server **c3po**. Démon pppd podporuje velké množství směrovacích schémat. Ty budou detailněji probrány dále v této kapitole.

8.4 Použití souborů options

Dříve, než démon pppd analyzuje argumenty příkazové řádky, projde několik souborů, zda neobsahují implicitní volby. Tyto soubory mohou obsahovat libovolné argumenty příkazové řádky, jež mohou být uvedeny na několika řádkách. Komentáře jsou uvozeny symbolem křížku.

Prvním souborem voleb je soubor s názvem /etc/ppp/options, který se při spuštění démona pppd vždy prohlíží. Je dobré využít tohoto souboru ke specifikaci některých globálních implicitních nastavení, čímž zabráníte vašim uživatelům v nechtěném poškození bezpečnosti celého systému. Chcete-li například, aby démon pppd vyžadoval od svého protějšku ně-

5 Implicitní směr je instalován pouze v případě, že ještě není přítomen žádný implicitní směr.

jaký typ ověření totožnosti (buď pomocí protokolu PAP, nebo pomocí protokolu CHAP), uveďte v tomto souboru volbu `auth`. Tuto volbu nemůže uživatel potlačit, takže nemůže navázat spojení pomocí protokolu PPP s žádným systémem, který není v ověřovací databázi.

Druhým souborem voleb, který se čte po souboru `/etc/ppp/options`, je soubor `.ppprc`. Nachází se v domovském adresáři příslušného uživatele. Každý uživatel tak může specifikovat svoji vlastní množinu implicitních voleb.

Vzorový soubor `/etc/ppp/options` by mohl vypadat asi takto:

```
# Globální volby pro pppd běžící na vlager.vbrew.com
auth                        # požaduj autorizaci
usehostname                 # pro CHAP použij lokální jméno
lock                        # používej zamykání UUCP
domain vbrew.com            # naše doména
```

První dvě volby se vztahují k procesu ověřování totožnosti a budou vysvětleny dále. Klíčové slovo `lock` nastaví démona `pppd` tak, aby vyhovoval standardní metodě UUCP, která definuje blokování zařízení. Podle tohoto standardu vytvoří každý proces, který přistupuje k sériovému zařízení, konkrétně k zařízení `/dev/cua3`, v dočasném adresáři UUCP zamykající soubor s názvem `LCK..cua3`, jehož přítomnost bude signalizovat, že je dané zařízení momentálně využíváno. To proto, aby jiné programy, například `minicom` nebo `uucico`, nemohly otevřít sériové zařízení v okamžiku, kdy ho používá protokol PPP.

Důvodem začlenění těchto voleb do globálního konfiguračního souboru je skutečnost, že uživatel nemůže výše uvedené volby potlačit a ty tím pádem poskytují solidní úroveň zabezpečení. Všimněte si však, že některé volby potlačit lze; příkladem budiž řetězec *connect*.

8.5 Vytáčení za pomoci programu chat

Jednou z věcí, která se vám mohla zdát v předchozím příkladu nepohodlná, je nutnost manuálně navázat spojení dříve, než se spustí démon `pppd`. Na rozdíl od nástroje `dip` nemá démon `pppd` svůj vlastní skriptový jazyk, který by umožňoval vytočení a přihlášení ke vzdálenému systému. Místo toho spoléhá na nějaký externí program nebo skript příkazového interpretu, který za něj tento úkol provede. Příkaz, který se má provést, lze předat démonu `pppd` pomocí volby příkazové řádky *connect*. Démon `pppd` přesměruje standardní vstup a výstup příkazu na sériovou linku. Pro tento účel existuje jeden užitečný program s názvem `expect`, jehož autorem je Don Libes. Obsahuje výkonný jazyk vycházející z jazyka Tcl, který byl navržen přesně pro tento druh aplikace.

Balík pppd obsahuje podobný program s názvem chat, který umožňuje zadat komunikační skript ve stylu UUCP. Komunikační skript je v podstatě složen ze střídající se sekvence očekávaných řetězců, které přijdou ze vzdáleného systému a z odpovědí, jež posíláme. Následuje typický výpis z komunikačního skriptu:

```
ogin: b1ff ssword: s3kr3t
```

Tento příkaz říká programu chat, aby vyčkal, dokud vzdálený systém nepošle výzvu k přihlášení, a potom mu poslal přihlašovací jméno **b1ff**. Čekáme pouze na řetězec ogin:, takže nebude vadit, když bude přihlašovací výzva začínat malým nebo velkým písmenem l, nebo když dorazí ve zkomolené formě. Následující řetězec je opět očekávaný řetězec, který pozdrží program chat, dokud nepřijde výzva k zadání hesla, potom odešle naše heslo jako odpověď.

To je v podstatě hlavní účel komunikačních skriptů. Kompletní skript sloužící k vytáčení serveru PPP by musel samozřejmě obsahovat patřičné příkazy pro modem. Předpokládejme, že váš modem rozumí množině příkazů standardu Hayes a že telefonní číslo vytáčeného serveru je 318 714. Kompletní vyvolání programu chat, které by provedlo spojení se serverem **c3po,** by potom vypadalo následovně:

```
$ chat -v '' ATZ OK ATDT318714 CONNECT '' ogin: ppp word: GaGariN
```

Podle definice musí být první řetězec očekávaný řetězec, ale protože nám modem nic neodpoví, dokud ho nepobídneme, musíme sdělit programu chat, aby první očekávaný řetězec přeskočil. Provedeme to tak, že tento řetězec zadáme jako prázdný řetězec. Pak pokračujeme odesláním příkazu ATZ, což je inicializační řetězec pro modemy kompatibilní se standardem Hayes a následně budeme čekat na odpověď (OK). Následující řetězec pošle vytáčecí program programu chat společně s telefonním číslem a jako odpověď očekává zprávu CONNECT. Za ní opět následuje prázdný řetězec. Protože v tuto chvíli nechceme nic posílat, čekáme spíše na výzvu k přihlášení. Zbytek komunikačního skriptu funguje naprosto shodně s výše popsaným postupem.

Parametr -v zajistí, že program chat bude veškeré aktivity zapisovat do souboru.[6]

Zadávání komunikačního skriptu na příkazové řádce s sebou nese jistá rizika, protože si uživatelé mohou prohlédnout příkazovou řádku daného procesu pomocí příkazu ps. Tomuto problému se vyhnete, když umístíte komunikační skript do souboru, řekněmě *dial-c3po*. Pomocí parametru -f, za kterým následuje název souboru, sdělíte programu chat, aby četl skript ze souboru místo z příkazové řádky. Kompletní spuštění démona pppd by mohlo nyní vypadat následovně:

6 Upravíte-li soubor syslog.conf z důvodu přesměrování těchto kontrolních zpráv do souboru, pak se ujistěte, že k tomuto souboru nemůže přistupovat každý, protože program chat implicitně zapisuje celý komunikační skript, tedy úplně vše včetně hesel.

```
# pppd connect "chat -f dial-c3po" /dev/cua3/ 38400 -detach \
        crtscts modem defaultroute
```

Kromě volby *connect* specifikující skript pro vytáčení jsme do příkazové řádky přidali ještě další dvě volby: `-detach` sdělí démonu `pppd`, aby se odpojil od konzoly a jako proces se přepnul na pozadí. Klíčové slovo `modem` sděluje démonu `pppd`, aby na sériovém zařízení provedl některé akce specifické pro modem, jako je zavěšení linky před a po volání. Pokud toto klíčové slovo nepoužijete, nebude démon `pppd` monitorovat linku DCD na sériovém portu, a proto nebude schopen detekovat neočekávané zavěšení na vzdálené straně.

Výše uvedené příklady byly poměrně jednoduché; program `chat` však umožňuje psát mnohem složitější komunikační skripty. Jednou z užitečných vlastností je určení řetězce, při kterém se komunikační skript zastaví s chybou. Typickými řetězci pro zastavení běhu skriptu jsou zprávy jako `BUSY` nebo `NO CARRIER`, které generuje váš modem v případě, že je volané číslo obsazené nebo vzdálená strana nezvedá telefon. Aby program `chat` rozpoznal tyto zprávy okamžitě, ne až po uplynutí určité doby, můžete je zadat na začátku skriptu pomocí klíčového slova `ABORT`:

```
$ chat -v ABORT BUSY ABORT 'NO CARRIER' '' ATZ OK ...
```

Podobným způsobem můžete ve skriptu změnit hodnotu čekací doby pomocí volby `TIMEOUT`. Podrobnosti získáte na stránkách manuálu programu `chat(8)`.

Někdy budete také potřebovat určitý druh podmíněného spouštění jednotlivých částí komunikačního skriptu. Když například neobdržíte od vzdáleného konce výzvu pro přihlášení, budete chtít poslat příkaz BREAK nebo znak návrat vozíku (CR). Toho docílíte přidáním podřetězce k očekávanému řetězci. Ten se bude skládat ze sekvence posílaných a očekávaných řetězců, stejně jako tomu bylo v celém skriptu. Tyto řetězce jsou odděleny znakem pomlčka. Podřetězec je spuštěn vždy, když očekávaný řetězec, ke kterému se daný podřetězec vztahuje, není přijat v zadaném čase. V níže uvedeném příkladu upravíme komunikační řetězec následujícím způsobem:

```
ogin:-BREAK-ogin: ppp ssword: GaGariN
```

Když program `chat` neuvidí výzvu pro přihlášení, která by měla přijít ze vzdáleného konce, spustí podřetězec, který nejdříve pošle příkaz BREAK a potom bude čekat na opětovnou výzvu pro přihlášení. Pokud se nyní tato výzva objeví, bude skript pokračovat běžným způsobem, v opačném případě skončí s chybou.

8.6 Ladění nastavení protokolu PPP

Démon `pppd` bude implicitně zapisovat veškeré varování a chybové zprávy do souboru *syslog démona*. Do souboru `syslog.conf` musíte přidat položku, která toto zapisování přesměruje do souboru, případně na konzolu, v opačném případě prostředky syslog tyto zprávy zruší. Následující položka způsobí, že budou všechny zprávy posílány do souboru `/var/log/ppp-log`:

```
daemon.*                    /var/log/ppp-log
```

Jestliže vaše nastavení nefunguje ihned, můžete na základě tohoto log-souboru poskytnout první indicii o tom, co je v nepořádku. Pokud vám ani to nepomůže, můžete také pomocí volby `debug` zapnout speciální ladicí výstup. Tato volba nařídí démonu `pppd`, aby zapisoval do prostředků `syslog` obsahy všech poslaných nebo přijatých řídicích paketů.

Konečně nejdrastičtějším způsobem je povolení ladicích informací na úrovni jádra operačního systému. To je možné provést spuštěním démona `pppd` s volbou `kdebug`. Za touto volbou následuje číselný argument, který je bitově orientovanou operací OR nad následujícími hodnotami: 1 pro obecné ladicí informace, 2 pro vytištění obsahu všech příchozích rámců protokolu HDLC a při hodnotě 4 vypíše ovladač veškeré odcházející rámce protokolu HDLC. Abyste mohli zachytávat ladicí zprávy jádra operačního systému, musíte buď spustit démona `syslogd`, který čte soubor `/proc/kmsg` nebo démona `klogd`. Oba tito démoni směrují ladící informace jádra do souboru `syslog`.

8.7 Volby pro konfiguraci protokolu IP

Protokol IPCP slouží k dohodnutí několika parametrů IP v době konfigurace spojení. Každá z komunikujících stran může poslat paket s požadavky na konfiguraci protokolu IPCP (IPCP Configuration Reguest), který obsahuje informace o implicitních hodnotách, jež se mají změnit a jaké mají být jejich hodnoty. Po přijetí tohoto paketu prozkoumá vzdálený konec střídavě každou z těchto voleb a buď ji přijme, nebo ji odmítne.

Démon `pppd` vám dává k dispozici velký prostor pro řízení parametrů protokolu IPCP, které se bude snažit dohodnout. Tyto parametry můžete nastavovat pomocí různých voleb příkazové řádky, které probereme později.

8.7.1 Výběr IP-adres

V předchozím příkladu nám démon `pppd` vytočil server **c3po** a navázal spojení IP. Nebyla provedeny žádná opatření, která by některému z obou konců přidělila konkrétní IP-adresu. Místo toho jsme použili adresu hostitele **vlager** jako místní IP-adresu a serveru **c3po** jsme po-

nechali jeho vlastní IP-adresu. Někdy je však užitečné mít kontrolu nad tím, která adresa bude použita na jednom nebo druhého konci spojení. Démon `pppd` poskytuje několik typů takovéto kontroly.

Chcete-li si vyžádat konkrétní adresy, stačí démonu `pppd` předat následující volbu:

```
local_addr:remote_addr
```

Parametry `local_addr` a `remote_addr` mohou být zapsány buď ve formě tečkové notace, nebo jako názvy hostitelů.[7] Tento příkaz způsobí, že se démon `pppd` pokusí použít první adresu jako svou vlastní IP-adresu a druhou adresu jako IP-adresu protějšku. Pokud protějšek v průběhu sjednávání parametrů pomocí protokolu IPCP některou z těchto adres odmítne, nedojde k řádnému spojení IP.[8]

Pokud chcete nastavit pouze místní adresu a přitom hodláte akceptovat libovolnou adresu protějšku, ponechte část `remote_addr` volnou. Chcete-li například, aby server **vlager** používal místo své vlastní IP-adresy adresu **130.83.4.27**, měli byste na příkazové řádce zadat `130.83.4.27:`. Podobně chcete-li nastavit pouze adresu vzdáleného počítače, ponechte volné pole `local_addr`. Démon `pppd` pak použije adresu sdruženou s názvem vašeho hostitele.

Některé servery PPP, které obsluhují velké množství klientů, přidělují IP-adresy dynamicky: adresy jsou systému přiděleny pouze v okamžiku volání a ihned po jeho odhlášení jsou opět uvolněny. Spojujete-li se s takovýmto typem serveru, musíte se ujistit, že démon `pppd` nevyžaduje od serveru žádnou konkrétní IP-adresu, ale přijme adresu, kterou po vás server požaduje. To znamená, že nemůžete použít argument `local_addr`. Kromě toho musíte použít volbu `noipdefault`, která sdělí démonu `pppd`, aby nepoužil místní adresu hostitele, ale počkal na přidělení IP-adresy od protějšího počítače.

8.7.2 Směrování přes spojení pomocí protokolu PPP

Jakmile je nastaveno síťové rozhraní, nastaví obvykle démon pouze hostitelské směrování ke svému protějšku. Je-li vzdálený hostitel v síti LAN, budete se jistě chtít spojit také s hostiteli, kteří jsou „za hranicemi“ vašeho protějšku; to znamená, že musí být nastaveno směrování sítí.

Už jsme si ukázali, že démona `pppd` je možné za pomoci volby `defaultroute` požádat, aby nastavil implicitní směr. Tato volba je velmi užitečná v případě, že se vámi vytočený server PPP bude chovat jako internetová brána.

7 Použijete-li v této volbě názvy hostitelů, bude to mít nepříznivé důsledky na ověření totožnosti pomocí protokolu CHAP. Podrobnosti najdete v níže uvedené stati o protokolu CHAP.

8 Protějšku s protokolem PPP můžete povolit potlačení vašich návrhů IP-adres tak, že démonu *pppd* předáte volby `ipcp-accept-local` a `ipcp-accept-remote`. Detaily získáte na stránkách manuálu.

Realizace obráceného případu, kdy se váš systém chová pro konkrétního hostitele jako brána, je také poměrně jednoduchá. Vězměte si například několik zaměstnanců ze společnosti Virtual Brewery, jejichž domovský počítač má název **loner**. Když se takový zaměstnanec spojí s hostitelem **vlager** pomocí protokolu PPP, použije ke spojení adresu podsítě virtuálního pivovaru. Na bráně **vlager** můžeme nyní použít volbu `proxyarp`, která nainstaluje pro hostitele **loner** proxy ARP. Takto bude hostitel **loner** automaticky dostupný ze všech hostitelů, kteří se nacházejí v síti pivovaru nebo vinárny.

Ne vždy to ale jde tak jednoduše, jako v tomto případě, příkladem budiž spojení dvou lokálních sítí. To obvykle vyžaduje přidání konkrétního síťového směrování, protože tyto sítě již mohou mít své vlastní implicitní směry. Kromě toho, když necháte u obou protějšků nastaven implicitní směr na spojení pomocí protokolu PPP, vznikne smyčka, ve které budou mezi oběma protějšky obíhat pakety určené pro neznámé lokace tak dlouho, dokud nevyprší jejich doba přežití (TTL).

Jako příklad předpokládejme, že si společnost Virtual Brewery otevře pobočku v nějakém jiném městě. Tato pobočka bude mít svůj vlastní Ethernet, který bude používat IP-adresu **191.72.3.0**, což je podsíť 3 v síti pivovaru. Síť pivovaru je třídy B. Lidé z pobočky se chtějí spojit s hlavním Ethernetem pivovaru pomocí spojení PPP, aby si mohli aktualizovat databáze zákazníků atd. Hostitel **vlager** se opět chová jako brána; jeho protějšek se nazývá **sub-etha** a má přidělenu IP-adresu **191.72.3.1**.

Když se hostitel **sub-etha** spojí s hostitelem **vlager**, nastaví jako obvykle implicitní směr na hostitele **vlager**. Nicméně na bráně **vlager** musíme nainstalovat síťové směrování pro podsíť 3, které bude ukazovat na hostitele **sub-etha**. K tomuto účelu použijeme ještě neprobranou vlastnost démona `pppd` – příkaz `ip-up`. Jedná se o skript příkazového interpretu nebo program umístěný v adresáři `/etc/ppp`, který je spouštěn po konfiguraci rozhraní PPP. Je-li tento program přítomný, spustí se s následujícími parametry:

```
ip-up iface device speed local_addr remote_addr
```

Argument `iface` označuje používané síťové rozhraní, parametr `device` představuje cestu k používanému sériovému zařízení (jsou-li použity parametry stdin/stdout, bude tato cesta ukazovat do souboru `/dev/tty`) a parametr `speed` představuje rychlost daného zařízení. Parametry `local_addr` a `remote_addr` předávají IP-adresy, které použijí oba počítače účastnící se spojení. Tyto adresy jsou uvedeny v tečkové notaci. V našem příkladu by měl skript `ip-up` obsahovat následující část kódu:

```
#!/bin/sh
case $5 in
191.72.3.1)             # to je sub-etha
```

```
        route add -net 191.72.3.0 gw 191.72.3.1;;
esac
exit 0
```

Jakmile je spojení pomocí protokolu PPP ukončeno, je podobným způsobem použit skript `/etc/ppp/ip-down`, který vrátí zpět všechny akce provedené skriptem `ip-up`.

Nicméně směrovací schéma ještě není kompletní. Na obou hostitelích s protokolem PPP jsme nastavili položky směrovací tabulky, avšak žádný z hostitelů obou sítí dosud neví nic o existujícím spojení pomocí protokolu PPP. To nebude problém v případě, že mají všichni hostitelé v pobočce vinárny nastaveno svůj implicitní směr na hostitele **sub-etha** a všichni hostitelé v síti pivovaru mají nastaven implicitní směrování na bránu **vlager**. Není-li to váš případ, budete zřejmě nuceni použít nějakého směrovacího démona, například démona *gated*. Jakmile na bráně **vlager** vytvoříte síťové směrování, bude směrovací démon vysílat nové směrování všem hostitelům, kteří jsou připojeni k podsítím.

8.8 Řídicí parametry spojení

V předcházejících statích jsme se setkali s protokolem LCP (Protokol na řízení spojení), který je používán pro sjednávání charakteristik spojení a k testování spojení.

Dvě nejdůležitější volby, jež mohou být sjednávány pomocí protokolu LCP, jsou maximální příjmová jednotka (MRU) a asynchronní řídicí mapa znaků (Asynchronous Control Character Map). Existuje i spousta dalších voleb, které je možné pomocí tohoto protokolu nastavovat, ty jsou však příliš specializované na to, abychom se zde jimi mohli zabývat. Jejich případný popis najdete v RFC 1548.

Asynchronní řídicí mapa znaků, které se hovorově říká asynchronní mapa, slouží u asynchronních spojení, jako jsou telefonní linky, k identifikaci řídicích znaků, které musí být vynechány (musí být nahrazeny konkrétní sekvencí dvou znaků). Měli byste se například vyvarovat použití znaků XON a XOFF, které jsou používány k softwarovému řízení toku dat a špatně nakonfigurovaný modem by se mohl při přijetí takového znaku zablokovat. Dalším kandidátem je kombinace kláves `Ctrl+]` (znak pro ukončení `telnetu`). Protokol PPP umožňuje vynechat libovolné znaky s ASCII kódy od 0 do 31, pokud jsou uvedeny v asynchronní mapě.

Asynchronní mapa je bitová mapa o šířce 32 bitů, kde nejnižší platný bit odpovídá ASCII znaku NUL a nejvyšší platný bit odpovídá ASCII znaku s hodnotou 31. Je-li příslušný bit nastaven, znamená to, že odpovídající znak musí být před odesláním po lince vynechán. Implicitně je asynchronní mapa nastavena na hodnotu *0xffffffff*, to znamená, že budou všechny řídicí znaky vynechány.

Chcete-li sdělit vašemu protějšku, že nemusí přeskakovat všechny řídicí znaky, ale jen některé z nich, můžete zadat novou asynchronní mapu, kterou předáte démonu `pppd` pomocí volby `asyncmap`. Mají-li se přeskočit pouze znaky `^S` a `^Q` (znaky s ASCII hodnotou 17 a 19, které se obecně používají pro řízení XON a XOFF), použijte následující volbu:

```
asyncmap 0x000A0000
```

Maximální příjmová jednotka neboli MRU signalizuje protějšku, jakou maximální velikost rámců protokolu HDLC chceme používat. I když vám to možná připomíná hodnotu MTU (maximální přenosová jednotka), nemají spolu tyto dvě hodnoty téměř nic společného. Hodnota MTU je parametrem pro síťové zařízení jádra operačního systému a popisuje maximální velikost rámce, o kterou se může příslušné rozhraní starat. Hodnota MRU říká vzdálenému konci, že nemá generovat rámce větší než je hodnota MRU; rozhraní však musí být bez ohledu na tuto skutečnost schopno přijmout rámečky až do velikosti 1 500 bajtů.

Volba hodnoty MRU proto není ani tak otázkou toho, co je daná linka schopna přenést, jako spíše otázkou nastavení, se kterým dosáhnete nejvyššího výkonu. Pokud hodláte po příslušném spojení provozovat interaktivní aplikace, pak je dobré nastavit hodnotu MRU přinejmenším na hodnotu 296 bajtů, aby nějaký větší paket (konkrétně například paket z relace FTP) nezpůsobil „skok“ vašeho kurzoru. Chcete-li démonu `pppd` sdělit, že hodláte používat MRU o velikosti 296 bajtů, měli byste mu tuto hodnotu předat pomocí parametru `mru 296`. Nicméně malé hodnoty MRU mají smysl pouze v případě, že nemáte zakázánu VJ kompresi hlaviček (implicitně je povolena).

Démon `pppd` rozumí také některým volbám protokolu LCP, které nastavují celkové chování procesu sjednávání parametrů, jako je maximální počet konfiguračních požadavků, které si mohou obě strany vyměnit, než se spojení ukončí. Dokud si nebudete absolutně jisti tím, co děláte, neměli byste tyto parametry měnit.

Kromě toho existují ještě dvě volby týkající se opakování echo zpráv protokolu LCP. Protokol PPP definuje dvě zprávy, opakování požadavku (Echo Request) a opakování odpovědi (Echo Response). Démon `pppd` používá tuto vlastnost ke kontrole funkčnosti spojení. Tuto vlastnost můžete povolit za pomoci volby `lcp-echo-interval`, za kterou následuje časový údaj v sekundách. Pokud nebudou v tomto intervalu ze vzdáleného hostitele přijaty žádné rámečky, vygeneruje démon `pppd` zprávu Echo Request a bude očekávat, že mu protějšek vrátí hlášku Echo Response. Jestliže se tak nestane, pak se spojení po určitém počtu odeslaných požadavků ukončí. Tento počet je možné nastavit pomocí volby `lcp-echo-failure`. Implicitně je tato volba zakázána.

8.9 Obecné úvahy nad bezpečností

Špatně nakonfigurovaný démon protokolu PPP může mít z hlediska bezpečnosti zhoubný vliv na celý systém. V nejhorším případě to vypadá tak, že má každý uživatel možnost připojení do vaší sítě Ethernet (a to je velice špatné). V této stati si povíme o několika málo opatřeních, které by měly zabezpečit vaši konfiguraci protokolu PPP.

Jedním z problémů démona `pppd` je to, že pro konfiguraci síťových zařízení a směrovací tabulky vyžaduje přístupová práva **superuživatele**. To obvykle vyřešíte tak, že ho necháte běžet setuid **root**. Nicméně démon `pppd` dovoluje uživatelům nastavovat z hlediska bezpečnosti různě významné volby. Abyste se chránili před útoky, které může uživatel spustit při manipulaci s těmito volbami, doporučuje se nastavit několik implicitních hodnot do souboru `/etc/ppp/options`, například ty, které jsme použili v ukázkovém souboru v sekci Použití souboru `options`. Některé z nich, například volby pro ověření totožnosti, nemůže uživatel potlačit, a proto poskytují dostatečnou ochranu před zneužitím.

Samozřejmě, že se musíte chránit i ze strany systémů, se kterými provozujete spojení na bázi protokolu PPP. Abyste se chránili před hostiteli, kteří se vydávají za někoho jiného, měli byste při komunikaci s protějškem vždy používat nějaký druh ověření totožnosti. Kromě toho byste měli cizím hostitelům zakázat používání IP-adres dle vlastního výběru a omezit jejich výběr jen na několik málo IP-adres. V následující stati se budeme zabývat právě těmito tématy.

8.10 Ověřování totožnosti pomocí protokolu PPP

8.10.1 Protokol CHAP versus protokol PAP

Pokud systém používá protokol PPP, může požadovat po svém protějšku, aby dokázal svoji totožnost pomocí jednoho ze dvou protokolů sloužících k ověřování totožnosti. Jsou to protokol pro ověření hesla (Password Authentication Protocol - PAP) a protokol pro ověření inicializační výzvy (Challenge Handshake Authentication Protocol – CHAP). Po navázání spojení může každá strana žádat po svém protějšku, aby dokázal svoji totožnost, bez ohledu na to, zda jde o volajícího nebo o volaného. V dalším výkladu budu v případech, kdy bude nutné, rozlišovat mezi ověřovaným systémem a ověřovatelem neurčité pojmy ‚klient' a ‚server'. Démon protokolu PPP může požádat svůj protějšek, aby dokázal svoji totožnost. To provede tak, že pomocí protokolu LCP pošle další konfigurační požadavek, v němž bude uveden požadovaný ověřovací protokol.

Protokol PAP pracuje v podstatě stejně jako klasická přihlašovací procedura. Klient ověří svoji totožnost tak, že serveru pošle jméno uživatele a (volitelně zašifrované) heslo, tato data server porovná se svou vlastní tajnou databází. Tuto techniku mohou obejít tzv. naslouchači, kteří se snaží získat potřebná hesla tak, že odposlouchávají sériovou linku a potom provádějí útoky systémem pokus omyl.

Protokol CHAP takové nedostatky nemá. Ověřovatel (tj. server) pošle klientovi náhodně vygenerovaný řetězec s „výzvou“ a spolu s ním i svůj název hostitele. Klient na základě názvu hostitele vyhledá příslušné tajné informace, zkombinuje je s přijatou výzvou a zašifruje tento řetězec pomocí jednosměrné šifrovací funkce. Výsledek pak společně s názvem hostitele klienta pošle zpět serveru. Server pak provede stejné výpočty a dojde-li k témuž výsledku, povolí klientovi přístup.

Další vlastností protokolu CHAP je, že nevyžaduje po klientovi ověření jeho totožnosti jen při spuštění, ale posílá výzvy v pravidelných intervalech, aby se ujistil, že klienta nenahradil nějaký vetřelec, například přepnutím telefonních linek.

Démon `pppd` implicitně nevyžaduje po svém protějšku ověření totožnosti, ale souhlasí se svým vlastním ověřením totožnosti v případě, že je o to vzdáleným počítačem požádán. Protože je protokol CHAP mnohem výkonnější než protokol PAP, snaží se ho démon `pppd` používat, kdykoliv jen je to možné. Pokud ho protějšek nepodporuje nebo pokud démon `pppd` nemůže pro vzdálený systém nalézt v souboru `chap-secrets` tajné informace protokolu CHAP, pak démon přepne na protokol PAP. Nenajde-li tajné informace pro svůj protějšek ani v protokolu PAP, odmítne prokázat svoji totožnost. V důsledku toho dojde k ukončení spojení.

Toto chování se může upravit několika způsoby. Použijete-li například klíčové slovo `auth`, bude démon `pppd` požadovat po svém protějšku, aby prokázal svou totožnost. Pro tento účel bude démon souhlasit s použitím protokolu CHAP nebo protokolu PAP, má-li ve své databázi CHAP, resp. PAP uloženy tajné informace pro svůj protějšek. Existují i další volby, kterými je možno zapnout nebo vypnout konkrétní protokol pro ověření totožnosti, avšak ty zde nebudeme rozebírat. Chcete-li o nich více podrobností, nahlédněte prosím do manuálové stránky démona `pppd(8)`.

Budou-li všechny systémy, se kterými máte spojení pomocí protokolu PPP, souhlasit s ověřením své totožnosti, měli byste vložit volbu `auth` do globálního souboru `/etc/ppp/options` a pro každý systém definovat v souboru `chap-secrets` příslušná hesla. Pokud daný systém protokol CHAP nepodporuje, vložte záznam o tomto systému do souboru `pap-secrets`. Pak si můžete být jisti, že se žádný neověřený systém k vašemu hostiteli nepřipojí.

Následující dvě stati budou probírat tajné soubory protokolu PPP, konkrétně soubory `pap-secrets` a `chap-secrets`. Najdete je v adresáři `/etc/ppp` a obsahují trojici klientů, serverů a hesel, volitelně následované seznamem IP-adres. Interpretace polí klienta a serveru je u protokolu CHAP jiná, než u protokolu PAP a závisí také na tom, zda dokazujeme svoji totožnost našemu protějšku, nebo zda požadujeme po serveru, aby nám svoji totožnost prokázal on.

8.10.2 Soubor Secrets protokolu CHAP

Když musíte nějakému serveru dokázat svoji totožnost pomocí protokolu CHAP, vyhledá démon `pppd` v souboru `chap-secrets` položku, která má pole klienta totožné s polem názvu místního hostitele a pole serveru bude mít totožné s názvem vzdáleného hostitele, který byl doručen pomocí výzvy protokolu CHAP. Když požadujete po protějšku, aby dokázal svoji totožnost, budou role obrácené: démon `pppd` vyhledá položku, u které se pole klienta shoduje s názvem vzdáleného hostitele (který je zaslán v odpovědi protokolu CHAP) a pole serveru je shodné s názvem místního hostitele.

Následuje vzorový soubor `chap-secrets` pro bránu **vlager**:[9]

```
# Soubor CHAP secrets pro vlager.vbrew.com
#
# klient          server           tajemství            adresa
#-------------------------------------------------------------------
vlager.vbrew.com  c3po.lucas.com   "Use The Source" vlager.vbrew.com
c3po.lucas.com    vlager.vbrew.com "riverrun, pasteve"  c3po.lucas.com
*                 vlager.vbrew.com "VeryStupidPassword" pub.vbrew.com
```

Při spojení pomocí protokolu PPP s hostitelem **c3po** požádá hostitel **c3po** bránu **vlager**, aby mu dokázala svoji totožnost pomocí protokolu CHAP. Brána to provede tak, že pošle výzvu protokolu CHAP. Poté démon `pppd` vyhledá v souboru `chap-secrets` položku, která má pole klienta shodné s adresou **vlager.vbrew.com** a její pole serveru se shoduje s adresou **c3po.lucas.com**;[10] v našem příkladu to bude položka na prvním řádku. Pak vytvoří démon `pppd` z řetězce obsahujícího výzvu a z tajných informací (`Use The Source`) odpověď pro protokol CHAP a pošle ji hostiteli **c3po**.

Ve stejném okamžiku sestaví démon `pppd` výzvu protokolu CHAP určenou pro hostitele **c3po**, která bude obsahovat jedinečný řetězec s výzvou společně s plně kvalifikovaným názvem hostitele **vlager.vbrew.com**. Hostitel **c3po** vytvoří výše popsaným způsobem odpověď pro protokol CHAP a tuto odpověď vrátí zpět bráně **vlager**. Nyní démon `pppd` vyjme z od-

[9] Uvozovky nejsou součástí hesla, slouží pouze k ochraně bílých mezer uvnitř hesla.

[10] Toto jméno je získáno z výzvy protokolu CHAP

povědi hostitelský název klienta (**c3po.vbrew.com**) a vyhledá v souboru `chap-secrets` řádek, v němž klientovi odpovídá hostitel **c3po** a serveru pak hostitel **vlager**. Této podmínce vyhovuje druhý řádek, takže démon `pppd` zkombinuje výzvu protokolu CHAP s tajnou informací `riverrun, pasteve`, zašifruje je a výsledek porovná s odpovědí protokolu CHAP, která přišla od hostitele **c3po**.

Volitelné čtvrté pole obsahuje seznam IP-adres, které jsou přijatelné pro klienty uvedené v prvním poli. Adresy mohou být zadány v tečkové notaci nebo jako názvy hostitelů, které vyhledá resolver. Požaduje-li například hostitel **c3po** během sjednávání spojení pomocí protokolu IPCP IP-adresu, která není uvedena v tomto seznamu, bude požadavek zamítnut a protokol IPCP bude ukončen. Proto je ve výše uvedeném vzorovém souboru hostitel **c3po** omezen pouze na použití své vlastní IP-adresy. Je-li pole s adresami prázdné, budou povoleny libovolné adresy; pokud je zde znak „–", nemůže daný klient použít žádnou IP-adresu.

Třetí řádek ve vzorovém souboru `chap-secrets` povoluje libovolnému hostiteli navázat spojení pomocí protokolu PPP s bránou **vlager**, protože hodnota * u pole klienta nebo serveru odpovídá libovolnému názvu hostitele. Jediným požadavkem na spojení je, že klient musí znát tajné informace a musí používat adresu **pub.vbrew.com**. Položky se zástupnými znaky představující názvy hostitelů se mohou v souboru s tajnými informacemi objevit kdekoliv, protože démon `pppd` bude vždy používat nejkonkrétnější položku, která se vztahuje k danému páru polí server/klient.

Dále bychom měli říci něco o způsobu, jakým démon `pppd` dospěje k názvům hostitelů v souboru tajných informací. Jak jsme si již dříve vysvětlili, název vzdáleného hostitele dodá vždy protějšek v paketu výzvy protokolu CHAP nebo v paketu odpovědi protokolu CHAP. Místní název hostitele bude implicitně odvozen z volání funkce *gethostname(2)*. Pokud jste nastavili název systému jako nekvalifikovaný název hostitele, pak musíte démonu `pppd` poskytnout název domény pomocí volby `domain`:

```
# pppd ...domain vbrew.com
```

Tato volba připojí k hostiteli **vlager** název domény pivovaru u všech aktivit, které se vztahují k ověřování totožnosti. Dalšími volbami, které upravují představu démona `pppd` o názvu místního hostitele, jsou volby `usehostname` a `name`. Když na příkazovou řádku zadáte místní IP-adresu pomocí volby „`local` : varremote“, kde parametr `local` je název hostitele místo adresy respektující tečkovou notaci, použije démon `pppd` tento zápis jako název místního hostitele. Více podrobností najdete na manuálové stránce `pppd(8)`.

8.10.3 Soubor Secrets protokolu PAP

Soubor s tajnými informacemi protokolu PAP je velmi podobný souboru, který využívá protokol CHAP. První dvě pole vždy obsahují jméno uživatele a název serveru; třetí pole obsahuje tajné informace protokolu PAP. Když vzdálený počítač pošle požadavek na ověření totožnosti, použije démon `pppd` položku, která má pole server shodné s názvem místního hostitele a pole se jménem uživatele má shodné se jménem uživatele, které je posláno v žádosti. V době, kdy démon `pppd` dokazuje svému protějšku svoji totožnost, vybere démon `pppd` ze souboru tajných informací tu položku, u níž se pole se jménem uživatele shoduje s místním uživatelským jménem a pole serveru se shoduje s názvem vzdáleného hostitele. Potom tyto tajné informace odešle.

Vzorový soubor tajných informací protokolu PAP může vypadat asi takto:

```
# /etc/ppp/pap-secrets
#
# uživatel    server       heslo              adresa
vlager-pap    c3po         cresspahl          vlager.vbrew.com
c3po          vlager       DonaldGNUth        c3po.lucas.com
```

První řádek používáme při rozhovoru s hostitelem **c3po**. Druhý řádek popisuje, jak nám může uživatel se jménem **c3po** dokázat svoji totožnost.

Název **vlager-pap** ve sloupci jedna představuje jméno uživatele, které pošleme hostiteli **c3po**. Démon `pppd` vezme implicitně název místního hostitele jako uživatelské jméno, ale pomocí volby `user` můžete také zadat jiné jméno uživatele.

Při výběru položky ze souboru `pap-secrets` musí démon `pppd` znát název vzdáleného hostitele. Protože neexistuje žádný způsob, jak by ho mohl zjistit, musíte mu ho předat na příkazové rádce pomocí volby `remotename`, za níž následuje název hostitele vašeho protějšku. Například, abychom mohli používat výše uvedenou položku pro dokázání naší totožnosti hostiteli **c3po**, musíme na příkazovou řádku démona `pppd` doplnit následující volbu:

```
# pppd ... remotename c3po user vlager-pap
```

Ve čtvrtém poli (a ve všech následujících polích) můžete zadat, jaké adresy může daný hostitel používat, je to stejné jako u souboru s tajnými informacemi protokolu CHAP. Protějšek pak může požadovat adresy pouze z tohoto seznamu. Ve vzorovém příkladu požadujeme, aby hostitel **c3po** používal svou skutečnou IP-adresu.

Všimněte si, že protokol PAP představuje poměrně slabou metodu na ověření totožnosti a proto se doporučuje, kdykoliv je to možné, používat místo něj protokol CHAP. Z toho důvodu zde nebudeme detailněji rozebírat protokol PAP. Popis některých dalších vlastností protokolu PAP najdete na manuálové stránce `pppd(8)`.

8.11 Konfigurace serveru PPP

Spuštění démona `pppd` jako serveru je pouze otázkou doplnění příslušných voleb do příkazové řádky. Teoreticky je nutné vytvořit speciální účet, konkrétně účet **ppp**, přidělit mu nějaký skript nebo program, který bude fungovat jako přihlašovací příkazový interpret, jenž spustí démona `pppd` s těmito volbami. Do souboru `/etc/passwd` byste mohli například přidat následující řádek:

```
ppp:*:500:200:Public PPP Account:/tmp:/etc/ppp/ppplogin
```

Je možné, že budete chtít používat jiná než výše uvedená uid a gid. Dále budete muset nastavit pro výše uvedený účet nějaké heslo pomocí příkazu `passwd`.

Pak by mohl skript `ppplogin` vypadat následovně:

```
#!/bin/sh
#  ppplogin – skript pro spuštění pppd při přihlášení
mesg n
stty -echo
exec pppd –detach silent modem crtscts
```

Příkaz `msg` zabrání tomu, aby mohli ostatní uživatelé zapisovat do tty, například pomocí příkazu `write`. Příkaz `stty` vypíná opakování znaků. To je nutné z toho důvodu, že jinak by se na protějším počítači opakovaly všechny znaky, které pošle. Nejdůležitější z výše uvedených voleb démona `pppd` je volba `–detach`, protože zabraňuje démonu `pppd` odpojení od tty. Pokud tuto volbu neuvedeme, přejde démon `pppd` do pozadí a tím pádem se ukončí i skript příkazového interpretu.. To by ve svém důsledku mohlo znamenat zavěšení sériové linky a ukončení spojení. Volba `silent` sděluje démonu `pppd`, aby před začátkem posílání paketů vyčkal na příchozí paket z volaného systému. Tato volba zamezí vypršení přenosové doby v případě, kdy je volající systém příliš pomalý při spouštění svého klienta PPP. Volba `modem` sděluje démonu `pppd`, aby kontroloval linku DTR, pomocí níž lze zjistit, zda protějšek neukončil spojení. Volba `crtscts` zapíná hardwarové řízení toku dat.

Kromě těchto voleb můžete vyžadovat určitý druh ověření totožnosti, například pomocí volby `auth` v příkazové řádce nebo v globálním souboru s volbami. Manuálové stránky se také zmiňují o speciálních volbách, které slouží k zapnutí nebo vypnutí protokolů na ověření totožnosti.

Kapitola

9

Různé síťové aplikace

Poté, co úspěšně nastavíte IP a resolver, musíte začít věnovat pozornost službám, které hodláte po síti poskytovat. Tato kapitola se zabývá konfiguracemi několika malých síťových aplikací, včetně serveru `inetd` a programů z rodiny `rlogin`. Krátce se také zmíníme o rozhraní RPC (Remote Procedure Call), na kterém jsou založeny služby NFS (Network File System) a NIS (Network Information System). Bohužel konfigurace systémů NFS a NIS vyžaduje trochu více prostoru, proto budou tyto systémy probrány v samostatných kapitolách. To platí i pro elektronickou poštu a pro síťové news.

Samozřejmě v této knize nemůžeme probrat všechny síťové aplikace. Budete-li si chtít nainstalovat nějakou z aplikací, která zde není uvedena, jako například programy `talk`, `gopher` nebo `Xmosaic`, nahlédněte prosím na stránky jejich manuálů, kde najdete více informací.

9.1 Super-server inetd

Služby jsou často poskytovány pomocí tzv. *démonů*. Démon je program, který otevře určitý port a čeká na příchozí spojení. Pokud k takovému spojení dojde, vytvoří proces potomka, jenž toto spojení přijme, zatímco rodičovský proces bude stále pokračovat v odposlouchávání dalších požadavků. Tento koncept má nevýhodu v tom, že každá poskytovaná služba musí mít spuštěného démona, který odposlouchává port z důvodu případného výskytu spojení, což obecně znamená, že se plýtvá systémovými zdroji, například odkládacím prostorem.

Proto většina unixových instalací spouští tzv. „super-server“, který vytvoří sockety pro většinu služeb, které současně odposlouchává pomocí systémového volání *select(2)*. Když vzdálený hostitel požádá o některou z nabízených služeb, super-server to zaznamená a pro daný port vytvoří speciální server.

Obecně používaný super-server se nazývá `inetd`, internetový démon (Internet Daemon). Je spouštěn při procesu zavádění systému a seznam služeb, které má spravovat, získává ze spouštěcího souboru s názvem `/etc/inetd.conf`. Kromě výše zmíněných volaných serverů existuje i spousta triviálních služeb, které provádí démon `inetd` sám. Tyto služby se nazývají *vnitřní služby*. Jednou z nich jsou služba `chargen`, která generuje řetězec znaků a služba `daytime`, která vrací systémový čas.

V tomto souboru se položka skládá z jednotlivých řádek, které jsou tvořeny následujícími poli:

```
service type protocol wait user server cmdline
```

Jednotlivá pole mají následující význam:

`service` Určuje název služby. Název služby musí být převeden na číslo portu pomocí vyhledání názvu služby v souboru `/etc/services`. Tento soubor bude popsán ve stati Soubory `services` a `protocols`, která následuje níže.

`type` Určuje typ socketu. Nabývá buď hodnoty `stream` (pro protokoly využívající vlastností spojení), nebo hodnoty `dgram` (pro protokoly založené na datagramech). Proto by měly služby založené na protokolu TCP používat vždy hodnotu `stream`, zatímco služby založené na protokolu UDP by měly vždy používat hodnotu `dgram`.

`protocol` Určuje název přenosového protokolu, který bude daná služba používat. Musí to být korektní název protokolu, který se nachází v souboru `protocols`. Bude vysvětleno dále.

`wait` Tato volba se vztahuje pouze na typ socketu `dgram`. Nabývá hodnoty `wait` nebo `nowait`. Je-li zadána volba `wait`, spustí démon `inetd` pro daný port vždy pouze jeden server. V opačném případě bude po spuštění serveru okamžitě pokračovat v odposlouchávání portu.

To je užitečné u „jednocestných“ (single-threaded) serverů, které budou číst všechny přicházející datagramy tak dlouho, dokud nepřestanou přicházet a potom se ukončí. Většina serverů RPC vyhovuje tomuto typu a tudíž by u nich měla být uvedena volba `wait`. Druhý typ serverů, tzv. „vícecestné servery“ (multi-threaded), umožňují současně spouštět neomezený počet instancí; to je používáno jen velmi zřídka. U těchto serverů by měla být volba `nowait`.

Sockety typu `stream` by měly vždy používat volbu `nowait`.

`user` Tato volba určuje přihlašovací uživatelské id, pod kterým bude daný proces spouštěn. Tímto uživatelem je často uživatel **root**, avšak některé služby mohou používat i jiné účty. Zde je velmi vhodné uplatňovat princip nejnižších práv, což znamená, že byste neměli příkaz spouštět na účtu s vyššími právy, pokud je spouštěný program pro svou správnou funkci nevyžaduje. Například server news NNTP běží s právy **news**, zatímco služby, které by mohly představovat bezpečnostní rizika (jako je služba `tftp` nebo `finger`) jsou často spouštěny s právy **nobody**.

`server` Určuje název plné cesty ke spouštěnému programu serveru. Vnitřní služby jsou označeny klíčovým slovem `internal`.

`cmdline` Tato volba určuje příkazovou řádku, která bude předána danému serveru. Tato volba obsahuje argument 0, který odpovídá názvu příkazu. Pokud se program nechová jinak při vyvolání pod jiným názvem, je tímto názvem příkazu obvykle vlastní název programu.

Toto pole je u interních služeb prázdné.

```
#
# inetd služby
ftp        stream tcp nowait root    /usr/sbin/ftpd     in.ftpd -l
telnet     stream tcp nowait root    /usr/sbin/telnetd in.telnetd -
       b/etc/issue
#finger     stream tcp nowait bin     /usr/sbin/fingerd in.fingerd
#tftp       dgram  udp wait    nobody /usr/sbin/tftpd    in.tftpd
#tftp       dgram  udp wait    nobody /usr/sbin/tftpd    in.tftpd /bo-
ot/diskless
login      stream tcp nowait root    /usr/sbin/rlogind in.rlogind
shell      stream tcp nowait root    /usr/sbin/rshd     in.rshd
exec       stream tcp nowait root    /usr/sbin/rexecd   in.rexecd
#
#        vnitřní služby inetd
#
daytime    stream tcp nowait root internal
daytime    dgram  udp nowait root internal
time       stream tcp nowait root internal
time       dgram  udp nowait root internal
echo       stream tcp nowait root internal
echo       dgram  udp nowait root internal
```

```
discard   stream tcp nowait root internal
discard   dgram  udp nowait root internal
chargen   stream tcp nowait root internal
chargen   dgram  udp nowait root internal
```

Obrázek 9.1
Vzorový soubor `/etc/inetd.conf`

Na obrázku 9.1 vidíte vzorový soubor `inetd.conf`. Služba `finger` je uvedená jako komentář, což znamená, že není dostupná. To je často z důvodů bezpečnostních, protože může být útočníky zneužita k získání jmen uživatelů ve vašem systému.

Také služba `tftp` je okomentována. Služba `tftp` implementuje tzv. *primitivní protokol pro přenos souborů* (*Primitive File Transfer Protocol*), který umožňuje z vašeho systému přenést libovolný okolnímu světu dostupný soubor, aniž by byla prováděna jakákoliv kontrola pomocí hesla apod. To je nepříjemné zejména u souboru `/etc/passwd` a ještě horší je to v případě, kdy nepoužíváte stínová hesla.

Protokol TFTP obecně používají bezdiskoví klienti a X-terminály ke stahování startovacího kódu ze zaváděcích serverů. Pokud potřebujete spouštět službu `tftpd` z tohoto důvodu, ujistěte se, že jste omezili její rozsah pouze na ty adresáře, ze kterých budou klienti získávat soubory. To provedete tak, že příkazové řádce služby `tftpd` přidáte názvy těchto adresářů. Výše uvedené demonstruje druhý řádek výpisu.

9.2 Prostředky na řízení přístupu

Protože přístup k počítačům prostřednictvím sítě přináší mnoho bezpečnostních rizik, byly navrženy aplikace, které chrání před několika typy útoků. Některé z těchto aplikací mohou obsahovat chyby (nejdrastičtěji to demonstruje červ RTM Internet) nebo nemusí umět rozlišovat mezi bezpečnými hostiteli, od kterých budou přijímány požadavky na konkrétní službu, a nebezpečnými hostiteli, jejichž požadavky by měly zamítnout. Již jsme se krátce zmínili o službách `finger` a `tftp`. Tyto služby budete asi chtít povolit pouze „důvěryhodným hostitelům“, což ale nelze pomocí standardního nastavení, při kterém super-server `inetd` poskytuje tuto službu buď všem klientům, anebo žádnému klientovi.

Pro tento účel je vhodný nástroj `tcpd`,[1] tzv. zástupce démonů. U služeb protokolu TCP, které chcete monitorovat nebo chránit, je tento démon volán místo programu serveru. Nástroj `tcpd` zapisuje požadavky do démona `syslog`, kontroluje, zda je vzdálený hostitel oprávněn

[1] Napsal ho Wietse Venema, jehož e-mailová adresa je **wietse@wzv.win.tue.nl**

používat danou službu a pouze v tom případě spustí skutečný program serveru. Všimněte si, že tento postup nefunguje u služeb založených na protokolu UDP.

Chcete-li například zastoupit démona `finger`, musíte v souboru `inetd.conf` změnit odpovídající řádku následujícím způsobem:

```
# zastoupení démona finger
finger   stream   tcp      nowait   root     /usr/sbin/tcpd    in.fingerd
```

Pokud nepřidáte žádné řízení přístupu, bude se tato úprava klientovi jevit stejně, jako obvyklé nastavení služby `finger`, s tou výjimkou, že veškeré požadavky budou zapisovány s prioritou `auth` do souboru `syslog`.

Řízení přístupu je implementováno za pomoci dvou souborů, které se jmenují `/etc/hosts.allow` a `/etc/hosts.deny`. Tyto soubory obsahují položky, které některým hostitelům povolují, resp. zakazují přístup k určitým službám. Když nástroj `tcpd` vyřizuje od klienta s názvem hostitele **biff.foobar.com** požadavek na použití určité služby, jako je služba `finger`, vyhledá v souborech `hosts.allow` a `hosts.deny` (v uvedeném pořadí) položky odpovídající jak požadované službě, tak i hostiteli klienta. Pokud je odpovídající položka nalezena v souboru `hosts.allow`, bude přístup povolen bez ohledu na položky v souboru `hosts.deny`. Pokud ale bude odpovídající položka nalezena v souboru `hosts.deny`, bude požadavek odmítnut a spojení se ukončí. Jestliže se ani v jednom souboru odpovídající položka nenajde, bude požadavek přijat.

Položky v souboru s přístupy vypadají následovně:

```
servicelist: hostlist [:shellcmd]
```

Pole `servicelist` obsahuje seznam názvů služeb ze souboru `/etc/services` nebo klíčové slovo *ALL*. Chcete-li zadat všechny služby kromě služeb `finger` a `tftp`, použijte řetězec „*ALL EXCEPT finger, tftp*“.

Pole `hostlist` obsahuje seznam názvů hostitelů nebo IP-adres, případně klíčová slova *ALL*, *LOCAL* nebo *UNKNOWN*. Klíčovému slovu *ALL* vyhovují všichni hostitelé, zatímco klíčovému slovu *LOCAL* vyhovují pouze názvy hostitelů, kteří neobsahují tečku.[2] Klíčovému slovu *UNKNOWN* odpovídají všichni hostitelé, u nichž selhalo vyhledávání jejich názvu nebo adresy. Název začínající tečkou vyhovuje všem hostitelům, jejichž doména je shodná s poskytnutým názvem. Například název domény **.foobar.com** vyhovuje hostitelům na adrese **biff.foobar.com**. Podobná opatření existují i pro síťové IP-adresy a pro čísla podsítí. Chcete-li více detailů, nahlédněte prosím do manuálové stránky *hosts_access(5)*.

2 Tečku obvykle neobsahují jména hostitelů ze souboru `/etc/hosts`.

Chcete-li zakázat přístup ke službám `finger` a `tftp` všem hostitelům s výjimkou místních hostitelů, vložte do souboru `/etc/hosts.deny` následující řádku a soubor `/etc/hosts.allow` ponechte prázdný:

```
in.tftpd, in.fingerd:  ALL EXCEPT LOCAL, .your.domain
```

Volitelné pole `shellcmd` může obsahovat příkaz rozhraní, jenž bude vykonán při splnění dané položky. To je užitečné při nastavování pastiček, které mohou odhalit potenciální útočníky:

```
in.ftpd: ALL EXCEPT LOCAL, .vbrew.com : \
       echo "request from %d@%h" >> /var/log/finger.log; \
       if [ %h != "vlager.vbrew.com" ]; then \
            finger -l @%h >> /var/log/finger.log \
fi
```

Nástroj `tcpd` nahradí argumenty *%h* a *%d* skutečným názvem hostitele klienta, resp. skutečným názvem služby. Chcete-li více detailů, nahlédněte prosím do manuálových stránek *hosts_access(5)*.

9.3 Soubory services a protocols

Čísla portů, na kterých jsou nabízeny jisté „standardní“ služby, jsou definovány v RFC „Assigned Numbers“. Aby mohly servery nebo klienti převádět názvy služeb na tato čísla, musí být alespoň část z tohoto seznamu uložena na každém hostiteli; tato část je uložena v souboru s názvem `/etc/services`. V tomto souboru má každá položka následující syntaxi:

```
service port/protocol [aliases]
```

Zde pole `service` definuje název služby, pole `port` definuje port, na kterém je daná služba nabízena, a pole `protokol` definuje typ používaného transportního protokolu. Obecně toto pole nabývá buď hodnoty `udp`, nebo hodnoty `tcp`. Službu je možné nabízet i pro více protokolů, stejně tak lze nabízet různé služby na stejném portu, pokud používají odlišné protokoly. Pole `aliases` umožňuje zadat alternativní názvy stejné služby.

Soubor `services`, který je dodáván společně se síťovým softwarem, nebudete muset obvykle měnit. Přesto zde uvádíme malý výpis z tohoto souboru:

```
# Soubor services:
#
# známé služby
echo          7/tcp                  # Echo
echo          7/udp                  #
discard       9/tcp  sink null       # Discard
discard       9/udp  sink null       #
daytime       13/tcp                 # Daytime
daytime       13/udp                 #
chargen       19/tcp  ttytst source  # Character Generator
chargen       19/udp  ttytst source  #
ftp-data      20/tcp                 # File Transfer Protocol (Data)
ftp           21/tcp                 # File Transfer Protocol (Contr
telnet        23/tcp                 # Virtual Terminal Protocol
smtp          25/tcp                 # Simple Mail Transfer Protocol
nntp         119/tcp  readnews       # Network News Transfer Protoco
#
# unixové služby
exec         512/tcp                 # BSD rexecd
biff         512/udp  comsat         # nová pošta
login        513/tcp                 # vzdálené přihlášení
who          513/udp  whod           # vzdálené who a update
shell        514/tcp  cmd            # vzdálené příkazy
syslog       514/udp                 # vzdálené protokolování
printer      515/tcp  spooler        # vzdálený tisk
route        520/udp  router routed  # směrovací protokol
```

Všimněte si, že například služba `echo` je poskytována na portu 7 jak pro protokol TCP, tak i pro protokol UDP a port 512 používají dvě odlišné služby, konkrétně démon COMSAT (který upozorňuje uživatele na nově došlou poštu, viz *xbiff(1x)*), jenž používá protokol UDP, a služba vzdáleného spouštění (*rexec(1)*), která používá protokol TCP.

Podobně jako soubor se službami potřebuje i síťová knihovna nějaký způsob, jakým by mohla přeložit názvy protokolů – například těch protokolů, které jsou použity v souboru `services` – na čísla protokolů, kterým by rozuměly IP-vrstvy ostatních hostitelů. To se provede vyhledáním daného názvu v souboru `/etc/protocols`. Ten obsahuje na každé řádce jednu položku. Každá položka obsahuje název protokolu a číslo sdružené s daným protokolem. Provádění změn v tomto souboru je ještě méně pravděpodobné, než zasahování do souboru `/etc/services`. Nyní následuje vzorový soubor `protocols`:

```
#
# Internetové (IP) protokoly
#
ip      0       IP      # internet protocol
icmp    1       ICMP    # internet control message protocol
igmp    2       IGMP    # internet group multicast protocol
tcp     6       TCP     # transmission control protocol
udp     17      UDP     # user datagram protocol
raw     255     RAW     # RAW IP interface
```

9.4 Vzdálené volání procedur – RPC

Balík RPC neboli *Vzdálené volání procedur* (*Remote Procedure Call*) poskytuje velmi obecný mechanismus pro aplikace typu klient-server. Balík RPC vyvinula firma Sun Microsystems a v podstatě se jedná o sbírku nástrojů a knihoven funkcí. Důležitou aplikací postavenou na balíku RPC je systém NFS, síťový souborový systém, a systém NIS, síťový informační systém. Oba tyto systémy budou probírány v následujících kapitolách.

Server RPC se skládá ze sady procedur, které si může klient volat tím způsobem, že pošle serveru požadavek RPC společně s parametry procedury. Server spustí jménem klienta požadovanou proceduru a existuje-li návratová hodnota, pošle ji zpět klientovi. Aby mohl být balík RPC nezávislý na typu hardwaru, jsou všechna data předávaná mezi klientem a serverem převedena vysílajícím počítačem do tzv. formátu *externího vyjádření dat* (*External Data Representation* – XDR) a přijímající počítač je převede zpět do místního vyjádření dat.

Někdy způsobí zdokonalení RPC aplikace nekompatibilní změny v rozhraní volání procedur. Samozřejmě, že prostá výměna serveru může způsobit spadnutí všech aplikací, které stále očekávají původní chování. Proto mají programy RPC přiděleny číslo své verze, obvykle se začíná hodnotou 1 a při každé nové verzi rozhraní RPC je tato hodnota zvýšena. Server může často současně nabízet několik verzí RPC; v takovém případě označí klient ve svém požadavku číslo verze, kterou chce ve své implementaci dané služby používat.

Vlastní síťová komunikace probíhající mezi servery RPC a jejich klienty je zvláštní. Server RPC nabízí jednu nebo více skupin procedur; každá množina procedur se nazývá *program* a je jednoznačně určena tzv. *číslem programu*. Seznam definující přiřazení názvů služeb k číslům programů je obvykle uložen v souboru `/etc/rpc`. Na obrázku 9.2 vidíte výpis z tohoto souboru.

```
#
# /etc/rpc - různé služby založené na RPC
#
portmapper          100000  portmap sunrpc
rstatd              100001  rstat rstat_svc rup perfmeter
rusersd             100002  rusers
nfs                 100003  nfsprog
ypserv              100004  ypprog
mountd              100005  mount showmount
ypbind              100007
walld               100008  rwall shutdown
ypasswss            100009  ypasswd
bootparam           100026
ypupdated           100028  ypupdate
```

Obrázek 9.2
Vzorový soubor `/etc/rpc`

U sítí na bázi protokolu TCP/IP čelili autoři balíku RPC problému, jak sdružit čísla programů s obecnými síťovými službami. Zvolili takovou variantu, kdy každý server poskytuje pro každý program a pro každou verzi programu jak port pro protokol TCP, tak i port pro protokol UDP. Obecně budou aplikace při posílání dat používat protokol UDP a protokol TCP budou používat pouze v případě, že se posílaná data nevejdou do jediného datagramu protokolu UDP.

Samozřejmě, že klienti musí mít možnost zjistit port, na který je namapováno dané číslo programu. V tomto případě by bylo použití konfiguračního souboru příliš nepružné; jelikož aplikace RPC nepoužívají vyhrazené porty, nemůžeme mít žádnou jistotu, že port původně určený pro použití naší databázovou aplikací nebude obsazen nějakým jiným procesem. Proto aplikace RPC vyberou některý z dostupných portů a zaregistrují ho pomocí démona, tzv. *mapovače portů* (*portmapper deamon*). Tento démon se chová jako zprostředkovatel mezi všemi servery RPC, které jsou spuštěny na daném počítači: klient, který chce kontaktovat službu s daným číslem programu, se nejdříve bude dotazovat mapovače portů umístěné na hostiteli serveru, ten mu pak vrátí čísla portů TCP a UDP, na kterých může být daná služba dosažitelná.

Tato metoda má konkrétní nevýhodu v tom, že zavádí princip jediného selhání, podobně jako to činí démon `inetd` u standardních služeb. Avšak tento případ je ještě horší, protože když selže démon mapující porty, ztratí se veškeré informace o portech RPC; to obvykle způsobí, že budete muset manuálně znovu nastartovat všechny servery RPC nebo budete muset znovu nastartovat celý počítač.

V Linuxu se program na mapování portů nazývá `rpc.portmap` a je umístěn v adresáři `/usr/sbin`. Kromě toho, že je třeba se ujistit, zda je démon mapující porty skutečně spouštěn ze souboru `rc.inet2`, není třeba žádná další konfigurace.

9.5 Konfigurace příkazů r

Ke spouštění příkazů na vzdálených hostitelích existuje spousta možností. Jsou to například příkazy `rlogin`, `rsh`, `rcp` a `rcmd`. Všechny tyto příkazy vyvolají na vzdáleném hostiteli příkazový interpret a umožní uživateli spouštět programy. Samozřejmě, že klient musí mít na hostiteli, na kterém hodlá spouštět příkazy, založený účet. Tedy všechny tyto příkazy provádějí ověření totožnosti. Klient obvykle sdělí serveru své přihlašovací uživatelské jméno a server okamžitě požádá o zadání hesla, které je ověřeno běžným způsobem.

Avšak u konkrétních uživatelů je někdy vhodné nevyžadovat ověření totožnosti. Když se například často přihlašujete k ostatním počítačům v lokální síti LAN, měli byste být do ní vpuštěni, aniž byste museli pokaždé zadávat své heslo.

Vypnutí ověřování totožnosti je vhodné pouze u malého počtu hostitelů, jejichž databáze hesel jsou synchronizovány, nebo u malého počtu privilegovaných uživatelů, kteří musí z administrativních důvodů přistupovat k mnoha počítačům. Kdykoliv chcete lidem povolit přihlášení k vašemu hostiteli bez zadání přihlašovacího id nebo hesla, ujistěte se, že náhodně nepovolíte přístup někomu jinému.

Vypnutí ověřování totožnosti u příkazů z rodiny *r* lze provést dvěma způsoby. První z nich je určen pro superuživatele, a umožňuje povolit přihlašování několika nebo všech uživatelů na několik nebo všechny hostitele (později uvedené případy jsou velmi nešťastné), aniž by byli dotazováni na heslo. Tento přístup řídí soubor, který se jmenuje `/etc/hosts.equiv`. Soubor obsahuje seznam hostitelů a jmen uživatelů, kteří jsou považováni za rovnocenné s uživateli místního hostitele. Druhý způsob je pro uživatele, kteří povolí dalším uživatelům z určitých hostitelů přístup ke svému účtu. Tito uživatelé mohou být vypsáni v souboru `.rhosts`, který se nachází v domovském adresáři daného uživatele. Z bezpečnostních důvodů musí být tento soubor vlastněn uživatelem nebo superuživatelem a tento soubor nesmí být symbolickým odkazem, v opačném případě bude ignorován.[3]

Když klient požádá o službu z rodiny *r*, vyhledá se nejprve jeho hostitel a jeho uživatelské jméno v souboru `/etc/hosts.equiv` a potom v souboru `.rhosts` uživatele, pod jehož účtem se chce daný uživatel přihlásit. Jako příklad předpokládejme, že uživatel **janet** pracuje

[3] V prostředí systému NFS ho můžete chránit pomocí hodnoty 444, protože superuživatel je často velice omezován v přístupu k souborům na discích, které jsou připojeny pomocí systému NFS.

na hostiteli **gauss** a pokouší se přihlásit k účtu uživatele **joe** na hostiteli **euler**. V následujícím výkladu budeme považovat uživatele Janet za uživatele *klienta* a uživatele Joe za *místního* uživatele. Nyní napíše uživatel Janet na hostiteli **gauss** příkaz:

```
$ rlogin -l joe euler
```

Server nejprve zkontroluje v souboru *hosts.equiv*[4], zda má být uživateli Janet povolen volný přístup, a pokud neuspěje, pokusí se tohoto uživatele vyhledat v souboru .rhosts, který se nachází v domovském adresáři uživatele **joe**.

Soubor hosts.equiv na hostiteli **euler** vypadá asi takto:

```
gauss
euler
-public
quark.physics.groucho.edu      andres
```

Každý záznam se skládá z názvu hostitele, za nímž může volitelně následovat jméno uživatele. Jestliže se na řádce objeví pouze samotný název hostitele, bude všem uživatelům daného hostitele umožněn přístup ke svým místním účtům bez jakékoliv kontroly. Ve výše uvedeném příkladu bude uživateli Janet povoleno přihlášení ke svému účtu s názvem **janet**, pokud se bude přihlašovat z hostitele **gauss,** a totéž bude platit i pro kteréhokoliv dalšího uživatele s výjimkou uživatele **root**. Pokud se ale bude chtít uživatel Janet přihlásit jako uživatel **joe**, bude obvyklým způsobem požádán o zadání hesla.

Je-li za názvem hostitele uvedeno jméno uživatele, jako je tomu na poslední řádce výše uvedeného příkladu, bude tomuto uživateli umožněn přístup ke *všem* účtům kromě účtu **root** bez hesla.

Před názvem hostitele může být uveden symbol minus, viz položka „**-public**“. V tomto případě požadujeme, aby hostitel **public** ověřoval totožnost všech účtů, bez ohledu na to, jaká práva mají uživatelé přidělena ve svých souborech .rhosts.

Formát souboru .rhosts je identický s formátem souboru hosts.equiv, ale jeho význam je poněkud odlišný. Podívejte se na soubor .rhosts uživatele Joe na hostiteli **euler**:

```
chomp.cs.groucho.edu
gauss       janet
```

4 Poznamenejte si, že pokud se někdo pokusí přihlásit jako uživatel **root**, není prohledáván soubor hosts.equiv

První položka povolí uživateli **joe** volný přístup v případě, že se přihlašuje z hostitele **chomp.cs.groucho.edu**. Tato položka neovlivní práva žádného dalšího účtu na hostiteli **euler** nebo **chomp**. Druhá položka je variací výše uvedeného, která spočívá v tom, že uživateli **janet** bude povolen volný přístup k účtu uživatele Joe, pokud se přihlásí z hostitele **gauss**.

Pamatujte, že název hostitele klienta je získán pomocí reverzního mapování adresy volajícího hostitele na název hostitele, takže tento postup nebude fungovat u hostitelů, které resolver nezná. Název hostitele klienta odpovídá názvu uvedenému v souborech *hosts*, pokud je splněn jeden z následujících případů:

- Kanonický název hostitele klienta (nikoliv jeho přezdívka) přesně odpovídá názvu hostitele, který je uveden v tomto souboru.
- Je-li název hostitele klienta plně kvalifikovaným doménovým jménem (jako například jménem, které vrátí resolver při spuštěném systému DNS) a neodpovídá-li přesně názvu hostitele, který je uveden v souboru *hosts*, bude porovnán s názvem hostitele rozšířeným o název místní domény.

Kapitola

10

Síťový informační systém (NIS)

Provozujete-li lokální síť, budete chtít poskytovat uživatelům takové prostředí, které by jim připadalo transparentní. Jedním z důležitých kroků, které vedou k tomuto cíli, je zajištění synchronizace důležitých dat, jako jsou informace o účtech uživatelů, mezi všemi hostiteli. Ukázali jsme si, že pro rozlišení názvů hostitelů existuje mocná a propracovaná služba, konkrétně systém DNS. Pro ostatní úkoly žádná taková speciální služba neexistuje. Kromě toho, pokud spravujete pouze malou lokální síť, která nemá žádné spojení s Internetem, nebude se mnoho administrátorů namáhat s nastavováním systému DNS.

Tato situace vedla k tomu, že firma Sun vyvinula systém NIS, *síťový informační systém*. Systém NIS poskytuje prostředky pro obecný přístup k databázím, které lze využít k šíření informací všem hostitelům ve vaší síti. Těmito informacemi mohou být například informace obsažené v souborech `passwd` a `group`. To umožňuje, aby síť vypadala jako jediný systém se stejnými účty na všech hostitelích. Podobným způsobem můžete používat systém NIS k šíření informací o hostitelích, které jsou uloženy v souboru `/etc/hosts`, na všechny počítače v síti.

Systém NIS je založen na balíku RPC a skládá se ze serveru, z knihovny na straně klienta a z několika administrativních nástrojů. Původně se systém NIS nazýval *Yellow pages* (*Žluté stránky*), neboli YP. Toto označení se ještě stále používá jako informativní odkaz na tuto službu. Na druhou stranu jsou Yellow pages ochrannou známkou společnosti British Telecom, která požadovala po firmě Sun, aby tento název přestala používat. I s odstupem času však zůstaly některé názvy zakořeněny a zkratka YP se proto stále používá jako prefix názvů většiny příkazů, které se vztahují k sytému NIS, například `ypserv`, `ypbind` apod.

Dnes je systém NIS dostupný snad ve všech verzích Unixu a existují dokonce i jeho volné implementace. Jednu z těchto implementací obsahuje balík BSD verze Net-2. Je odvozena z veřejně dostupné implementace, kterou poskytla firma Sun. Již dlouhou dobu je kód knihovny klienta z této verze umístěn v knihovně GNU *libc*, zatímco administrativní nástroje teprve ne-

dávno portoval na platformu Linuxu pan Swen Thümmler.[1] Server systému NIS nemá žádnou referenční implementaci. Tobias Reber napsal další balík systému NIS, který obsahuje všechny nástroje i server; tento balík se nazývá `yps`.[2]

V současné době dokončil Peter Eriksson[3] kompletní přepis kódu systému NIS pod názvem NYS, který podporuje jak holý systém NIS, tak i mnohem dokonalejší systém NIS+ od firmy Sun. Balík NYS neposkytuje pouze množinu nástrojů a server, ale přidává do knihovny také celou novou sadu funkcí, které se časem zřejmě stanou součástí standardní knihovny *libc*. Tyto funkce obsahují nové konfigurační schéma pro rozlišování názvů hostitelů, které nahrazuje současné schéma používající soubor `host.conf`. Přínosy těchto funkcí budou rozebrány níže.

Tato kapitola se bude více soustřeďovat na balík NYS, než na další dva konkurenční balíky, které budeme označovat jako tzv. „tradiční“ kód systému NIS. Chcete-li pracovat s některým z těchto balíků, pak vám možná nebudou informace obsažené v této kapitole stačit. Chcete-li o nich získat více informací, sežeňte si prosím knihu o systému NIS, například publikaci *NFS and NIS* od Hala Sterna (viz [Stern92]).

Balík NYS se v současné době stále ještě vyvíjí, a proto standardní linuxové utility, jako jsou síťové programy nebo program `login`, zatím neovládají konfigurační schéma balíku NYS. Než se balík NYS stane součástí hlavní knihovny *libc*, budete si muset všechny tyto binární soubory aplikací sami překompilovat, aby byly schopny podporovat balík NYS. Při kompilaci libovolné z těchto aplikací pomocí programu *Makefile* zadejte sestavovacímu programu jako poslední parametr před knihovnou *libc* parametr `-lnsl`. Tento parametr vloží na místo standardních funkcí z knihovny C odpovídající funkce z knihovny *libnsl*, knihovny NYS.

10.1 Seznámení se systémem NIS

Systém NIS uchovává databázové informace v tzv. *mapách*, které obsahují páry klíčových hodnot. Mapy jsou uloženy na centrálním hostiteli, na němž běží systém NIS a z něhož mohou klienti získávat informace pomocí různých volání procedur RPC. Poměrně často bývají mapy uloženy v souborech DBM.[4]

---

1 Tento pán je k zastižení na adrese **swen@uni-paderborn.de**. Klienti systému NIS jsou dostupní v souboru s názvem `yp-linux.tar.gz` na adrese **sunsite.unc.edu** v adresáři `system/Network`.

2 Aktuální verze (v době psaní této knihy) byla `yps-0.21` a získáte ji na adrese **ftp.lysator.liu.se** v adresáři `/pub/NYS`.

3 Je k zastižení na adrese **pen@lysator.liu.se**.

4 DBM je jednoduchá knihovna pro správu databáze, která ke zrychlení vyhledávacích operací využívá transformačních technik. Z projektu GNU je k dispozici volně šiřitelná implementace DBM s názvem `gdbm`, která je součástí většiny distribucí Linuxu.

Vlastní mapy jsou obvykle generovány z hlavních textových souborů, jako jsou například soubory `/etc/hosts` nebo `/etc/passwd`. Pro některé soubory se vytvoří několik map, pro každý typ vyhledávacího klíče se vždy vytvoří jedna mapa. Například v souboru `hosts` můžete hledat jak název hostitele, tak i IP-adresu. Z tohoto souboru budou takto odvozeny dvě mapy, které se budou nazývat `hosts.byname`, resp. `hosts.byaddr`. Tabulka 10.1 uvádí běžně se vyskytující mapy společně se soubory, ze kterých byly vygenerovány.

Hlavní soubor	Mapa (Mapy)	
`/etc/hosts`	`hosts.byname`	`hosts.byaddr`
`/etc/networks`	`networks.byname`	`networks.byaddr`
`/etc/passwd`	`passwd.byname`	`passwd.byuid`
`/etc/group`	`group.byname`	`group.bygid`
`/etc/services`	`services.byname`	`services.bynumber`
`/etc/rpc`	`rpc.byname`	`rpc.bynumber`
`/etc/protocols`	`protocols.byname`	`protocols.bynumber`
`/usr/lib/aliases`	`mail.aliases`	

Tabulka 10.1
Některé standardní mapy systému NIS a jim odpovídající soubory

Možná se setkáte s tím, že některé balíky systému NIS nebo některé další programy podporují i jiné soubory a mapy. Tyto soubory a mapy mohou obsahovat informace o aplikacích, které nejsou v této knize probírány (například mapa `bootparams`) a mohou využívat některé servery BOOTP, anebo tyto soubory a mapy nemají v současné době v Linuxu žádnou funkci (příkladem jsou mapy `ethers.byname` nebo `ethers.byaddr`).

Pro některé mapy lidé obecně používají *přezdívky*, které jsou kratší a z tohoto důvodu se lépe píší. Chcete-li získat kompletní seznam přezdívek, kterým vaše nástroje systému NIS rozumí, spusťte následující příkaz:

```
$ ypcat -x
NIS map nickname translation table:
       "passwd" -> "passwd.byname"
       "group" -> "group.byname"
       "networks" -> "networks.byname"
       "hosts" -> "hosts.byname"
       "protocols" -> "protocols.byname"
       "services" -> "services.byname"
```

```
"aliases" -> "aliases.byname"
"ethers" -> "ethers.byname"
"rpc" -> "rpc.byname"
"netmasks" -> "netmasks.byname"
"publickey" -> "publickey.byname"
"netid" -> "netid.byname"
"passwd.adjunct" -> "passwd.adjunct.byname"
"group.adjunct" -> "group.adjunct.byname"
"timezone" -> "timezone.byname"
```

Server NIS se tradičně nazývá `ypserv`. Pro protřeby průměrné sítě zpravidla postačuje jediný server; rozsáhlé sítě možná dají přednost provozování několika těchto serverů na různých počítačích, čímž různé segmenty sítě odlehčí celkové zatížení serverových počítačů a směrovačů. Tyto servery jsou vzájemně synchronizovány tak, že některý z těchto serverů je nastaven jako *řídící server* a ostatní servery jsou nastaveny jako *řízené servery*. Mapy budou vytvořeny pouze na hostiteli s řídícím serverem. Odtud jsou distribuovány na všechny řízené servery.

Možná jste si všimli, že po celou dobu jsme o „sítích“ hovořili značně neurčitě; samozřejmě, že v systému NIS existuje přesný koncept týkající se dané sítě, kterou tvoří skupina všech hostitelů sdílejících pomocí systému NIS část jejích systémových konfiguračních dat: tímto konceptem je *doména* systému NIS. Bohužel domény systému NIS nemají nic společného s doménami systému DNS, s nimiž jsme se již setkali. Abychom se v této kapitole vyhnuli dvojsmyslnostem, bude vždy uváděn typ příslušné domény.

Domény systému NIS nabízí jen velmi slabé administrativní funkce. Ty jsou, kromě sdílení hesel mezi všemi počítači příslušné domény, většinou pro uživatele neviditelné. Proto má název poskytnutý doméně systému NIS význam pouze pro správce sítě. Zpravidla bude fungovat jakýkoliv název, který bude odlišný od všech názvů domén systému NIS vyskytujících se ve vaší lokální síti. Například správci ve společnosti Virtual Brewery mohou vytvořit dvě domény systému NIS, jednu z nich pro vlastní společnost Virtual Brewery a jednu pro společnost Virtual Winery, kterým přiřadí názvy **brewery**, resp. **winery**. Dalším poměrně běžným schématem je použití stejného názvu jak pro doménu systému DNS, tak i pro doménu systému NIS. K nastavení a zobrazení názvu domény systému NIS na vašem hostiteli můžete použít příkaz `domainname`. Když tento příkaz vyvoláte bez argumentů, zobrazí název aktuální domény systému NIS; budete-li chtít nastavit název domény, přihlašte se jako superuživatel a napište:

```
# domainname brewery
```

Domény NIS určují, na který server se budou aplikace dotazovat. Například program *login* se na hostiteli ve Virtual Winery může dotazovat na informace o uživatelských heslech pouze serveru NIS v této společnosti (nebo jednoho ze serverů v této společnosti, pokud je jich více); zatímco aplikace na hostiteli ve Virtual Brewaery může komunikovat pouze se serverem v této společnosti.

Teď ještě zbývá vyřešit jednu záhadu, konkrétně jak klient zjistí, se kterým serverem se má spojit. Nejjednodušší by bylo použít konfigurační soubor, v němž bude uveden hostitel, na kterém se server má hledat. Avšak tento přístup je poměrně nepružný, protože neumožňuje klientům používat různé servery (samozřejmě z téže domény) v závislosti na jejich dostupnosti. Proto spoléhají tradiční implementace systému NIS na speciálního démona nazývaného `ypbind`, který detekuje vhodný server NIS v jejich doméně systému NIS. Dříve, než může nějaká aplikace předat systému NIS jakýkoliv dotaz, si musí nejprve od démona `ypbind` zjistit, jaký server má k tomuto účelu použít.

Démon `ypbind` prozkoumává servery za pomoci vysílání do místní sítě IP; první, kdo odpoví, je považován za potenciálně nejrychlejší server, a proto bude použit pro všechny následující dotazy systému NIS. Po uplynutí určité doby nebo dojde-li k přerušení spojení se serverem, začne démon `ypbind` znovu prozkoumávat aktivní servery.

A nyní se dostáváme ke spornému bodu použití dynamické vazby, který spočívá v tom, že ji budete potřebovat pouze zřídka a její použití s sebou navíc přináší jisté bezpečnostní problémy: Démon `ypbind` slepě uvěří každému, kdo mu odpoví, což může být jak prostý server NIS, tak i zlomyslný vetřelec. Netřeba ani říkat, že tato vlastnost je zvláště nepříjemná, spravujete-li pomocí systému NIS své databáze s hesly. Abyste tomuto předešli, nepoužívá balík NYS implicitně démona `ypbind`, ale název hostitele serveru přebírá z konfiguračního souboru.

10.2 Systém NIS versus systém NIS+

Systémy NIS a NIS+ toho mají společného více, než jen svůj název a cíl. Systém NIS+ je strukturován zcela odlišným způsobem. Místo přímého jmenného prostoru s oddělenými doménami systému NIS používá hierarchický prostor s názvy, který je podobný jmennému prostoru systému DNS. Místo map jsou používány tzv. *tabulky*, které jsou složeny z řádků a sloupců, kde každý řádek reprezentuje objekt databáze systému NIS+, zatímco sloupce obsahují vlastnosti objektů, které systém NIS+ zná a o něž se stará. Každá tabulka příslušné domény systému NIS+ v sobě zahrnuje i tabulky svých rodičovských domén. Mimoto může položka v tabulce obsahovat spojení na další tabulku. Tyto vlastnosti umožňují strukturovat informace mnoha způsoby.

Tradiční systém NIS má číslo verze balíku RPC rovno 2, zatímco systém NIS+ používá verzi 3.

Zatím to vypadá tak, že systém NIS+ není příliš používán a ani já toho vlastně o něm příliš nevím. (Ve skutečnosti o něm nevím skoro nic.) Proto ho zde nebudeme rozebírat. Pokud se o tento systém zajímáte hlouběji, nahlédněte prosím do administrativního manuálu systému NIS+ od firmy Sun ([NISPlus]).

10.3 Systém NIS na straně klienta

Ovládáte-li psaní a portování síťových aplikací, pak si jistě všimnete, že většina výše uvedených map systému NIS odpovídá funkcím, které jsou obsaženy v knihovně C. Chcete-li například získat informace ze souboru `passwd`, budete k tomuto účelu běžně používat funkce *getpwnam(3)* a *getpwuid(3)*, které vrací informace o účtu sdružené s příslušným jménem uživatele, resp. s číselným id uživatele. Za normálních okolností provedou tyto funkce požadované vyhledání ve standardním souboru, což je soubor `/etc/passwd`.

Avšak implementace těchto funkcí v systému NIS toto chování upraví a vloží takové volání procedury RPC, aby server NIS, vyhledal příslušné uživatelské jméno nebo jeho id. Toto chování bude pro aplikaci zcela transparentní. Funkce může buď k danému souboru „připojit“ mapu systému NIS nebo může touto mapou původní soubor „nahradit“. Neznamená to samozřejmě skutečnou úpravu daného souboru, pouze to znamená, že se aplikaci bude takový soubor jevit, jako by k němu byla daná mapa připojena nebo jakoby byl touto mapou nahrazen.

V tradičních implementacích systému NIS existovala jistá pravidla týkající se map, které mají nahrazovat původní informace a které se mají připojovat k původním informacím. Některé z těchto map, například mapy ze skupiny `passwd`, vyžadovaly úpravy souboru `passwd`, které mohly při nesprávném postupu dát vzniknout bezpečnostním dírám. Aby se tomu zabránilo, používá balík NYS obecné konfigurační schéma, které určuje, zda bude konkrétní množina funkcí klienta používat původní soubory systému NIS nebo systému NIS+ a v jakém pořadí. Toto konfigurační schéma bude náplní dalších statí této kapitoly.

10.4 Provozování serveru NIS

Po přehršli teoretických technických informacích přišel čas věnovat se skutečné konfigurační praci. V této stati si probereme konfiguraci serveru NIS. Pokud už ve vaší síti běží nějaký server NIS, nebudete zřejmě chtít nakonfigurovat další vlastní server; v tom případě můžete tuto stať přeskočit.

Při experimentování se serverem se nezapomeňte ujistit, že jste mu nepřiřadili název domény systému NIS, který je již ve vaší síti používán. To by totiž mohlo narušit všechny síťové služby a rozzlobit spoustu lidí.

V současné době jsou v Linuxu volně dostupné dva servery NIS, jeden je obsažen v balíku `yps` od Tobiase Rebera a druhý v balíku `ypserv` od Petera Erikssona. Nemělo by záležet na tom, který z těchto balíků si pro provozování zvolíte, ani na tom, zda budete používat balík NYS nebo standardní kód klienta NIS, který je v současnosti součástí knihovny *libc*. Při psaní této knihy se zdálo, že kód pro správu řízených serverů NIS je dokonalejší v balíku `yps`. Tedy pokud budete muset používat řízené servery, bude možná vhodnější použít balík `yps`.

Po nainstalování programu serveru (`ypserv`) do adresáře `/usr/sbin` byste měli vytvořit adresář, v němž budou uloženy soubory map, které bude váš server distribuovat. Bude-li nastavena doména systému NIS na název domény **brewery,** měly by být mapy umístěny v adresáři `/var/yp/brewery`. Server zjistí, zda spravuje konkrétní doménu systému NIS tak, že ověří přítomnost adresáře s mapami. Pokud zakážete službu některé domény NIS, ujistěte se, že jste odstranili také odpovídající adresář.

Mapy jsou zpravidla kvůli rychlejšímu vyhledávání uloženy v souborech DBM. Tyto soubory jsou vytvořeny z hlavních souborů za pomoci speciálního programu s názvem `makedbm` (pro server od Tobiase) nebo `dbmload` (pro server od Petera). Tyto programy se nesmí zaměňovat. Převod hlavního souboru do formy analyzovatelné programem `dbmload` obvykle vyžaduje jistý trik typu `awk` nebo `sed`, který je poměrně obtížné popsat a navíc je hůře zapamatovatelný. Proto obsahuje balík `ypserv` od Petera Erikssona program pro vytvoření tohoto souboru (nazvaný `ypMakefile`), který všechnu potřebnou práci udělá za vás. Měli byste ho nainstalovat jako soubor `Makefile` do svého adresáře s mapami a upravit ho tak, aby obsahoval vámi distribuované mapy. Směrem k začátku souboru naleznete položku *all*, ve které jsou uvedeny služby nabízené programem `ypserv`. Tento řádek vypadá zhruba následovně:

```
all: ethers hosts networks protocols rpc services passwd group netid
```

Pokud například nechcete vytvořit mapy `ethers.byname` a `ethers.byaddr`, odstraňte z výše uvedené položky volbu `ethers`. Chcete-li si své nastavení otestovat, budou vám stačit pouze jedna nebo dvě mapy, například mapy `services.*`.

Po úpravě souboru `Makefile` napište v adresáři s mapami příkaz `make`. Tento příkaz požadované mapy automaticky vygeneruje a nainstaluje. Je třeba zajistit, aby se při každé změně hlavních souborů aktualizovaly i soubory s mapami, v opačném případě by síť provedené změny nezjistila.

Další stať popisuje, jak nakonfigurovat kód klienta systému NIS. Pokud vaše nastavení nefunguje, pak je třeba zjistit, zda na váš server přichází nějaké požadavky. Pokud serveru z balíku NYS zadáte jako příznak příkazové řádky `-d`, budou se na konzole vypisovat všechny příchozí dotazy systému NIS včetně vrácených odpovědí. Takto možná zjistíte v čem je kámen úrazu. Server od Tobiase žádnou takovou volbou nedisponuje.

10.5 Nastavení klienta systému NIS pomocí balíku NYS

Až do konce této kapitoly se budeme zabývat konfigurací klienta systému NIS.

Nejprve byste měli sdělit balíku NYS, který server se bude pro službu NIS používat, což provedete za pomoci konfiguračního souboru `/etc/yp.conf`. Velice jednoduchý vzorový soubor pro hostitele v síti Virtual Winery by mohl vypadat asi takto:

```
# yp.conf - konfigurační soubor pro systém NIS
#
domainname winery
server vbardolino
```

První příkaz sdělí všem klientům systému NIS, že jsou přiřazeni k doméně systému NIS s názvem **winery**. Pokud na tento řádek zapomenete, použije balík NYS název domény, který jste vašemu systému přidělili za pomoci příkazu `domainname`. Příkaz `server` označuje server NIS, který se bude používat. Samozřejmě, že v souboru `hosts` musí být nastavena IP-adresa odpovídající názvu hostitele **vbardolino**; popřípadě můžete v příkazu `server` použít vlastní IP-adresu.

Ve výše uvedeném příkladu sděluje příkaz `server` balíku NYS, aby použil označený server pokaždé, když je dostupná aktuální doména systému NIS. Pokud však svůj počítač často přemísťujete v rámci různých domén systému NIS, budete možná chtít mít v souboru `yp.conf` informace o několika doménách. V souboru `yp.conf` můžete mít informace o serverech pro různé domény systému NIS. Stačí pouze přidat k příkazu `server` název požadované domény systému NIS. Například pro laptop můžete výše uvedený vzor změnit následujícím způsobem:

```
# yp.conf - konfigurační soubor pro systém NIS
#
server vbardolino winery
server vstout     brewery
```

Takto upravený soubor `yp.conf` vám umožní přenášet laptop mezi libovolnými dvěma doménami. Při procesu zavádění systému pouze stačí příkazem `domainname` nastavit požadovanou doménu systému NIS.

Po vytvoření tohoto základního konfiguračního souboru a ujištění se, že je tento soubor všem dostupný, byste měli spustit svůj první test a zjistit, zda se můžete spojit se svým serverem. Vyberte si nějakou serverem distribuovanou mapu, například `hosts.byname`, a pokuste se ji získat pomocí utility `ypcat`. Nástroj `ypcat` by se měl, stejně jako všechny ostatní administrativní nástroje, nacházet v adresáři `/usr/sbin`.

```
# ypcat hosts.byname
191.72.2.2          vbeaujolais     vbeaujolais.linus.lxnet.org
191.72.2.3          vbardolino      vbardolino.linus.lxnet.org
191.72.1.1          vlager          vlager.linus.lxnet.org
191.72.2.1          vlager          vlager.linus.lxnet.org
191.72.1.2          vstout          vstout.linus.lxnet.org
191.72.1.3          vale            vale.linus.lxnet.org
191.72.2.4          vchianti        vchianti.linus.lxnet.org
```

Získaný výstup by měl vypadat podobně jako výše uvedený výpis. Pokud místo tohoto obdržíte chybovou zprávu „`Can't bind to server which serves domain`“ nebo nějakou podobnou zprávu, pak buď vámi zadaný název domény systému NIS nemá v souboru `yp.conf` definovaný odpovídající server, nebo je server z nějakého důvodu nedostupný. V druhém případě se ujistěte, že příkaz `ping` směrovaný na příslušného hostitele vrátí přijatelné výsledky, a že daný hostitel má skutečně spuštěn server NIS. Přítomnost serveru NIS na daném hostiteli můžete ověřit pomocí příkazu `rpcinfo`, který by měl vrátit následující výstup:

```
# rpcinfo -u serverhost ypserv
program 100004 version 2 ready and waiting
```

10.6 Výběr vhodných map

Abyste zajistili dosažitelnost daného serveru NIS, musíte rozhodnout, které konfigurační soubory nahradíte mapami systému NIS a ke kterým konfiguračním souborům mapy systému NIS připojíte. Mapy systému NIS budete zpravidla používat u funkcí, které vyhledávají hostitele a hesla. Funkce vyhledávající hostitele mají význam tehdy, pokud nemáte spuštěnu službu BIND. Funkce vyhledávající hesla povolují všem uživatelům přihlášení ke svému účtu z libovolného systému v příslušné doméně systému NIS; tato funkce obvykle vyžaduje sdílení centrálního adresáře `/home` pomocí systému NFS mezi všema hostiteli. Tato funkce je podrobně probrána v následující stati. Ostatní mapy, jako například mapa `service.byname`, nejsou příliš užitečné, ale mohou vám ušetřit část konfiguračních prací v případě, kdy instalujete nějaké síťové aplikace, které používají název služby, jenž není uveden ve standardním souboru `services`.

V případech, kdy vyhledávací funkce používá místní soubory nebo kdy se dotazuje serveru NIS, si budete zpravidla chtít zachovat určitou možnost volby. Balík NYS dovoluje nastavit pořadí, ve kterém daná funkce k těmto službám přistupuje. Toto pořadí je řízeno konfiguračním souborem `/etc/nsswitch.conf`, který nahrazuje *přepínač názvových služeb* (*Name*

Service Switch). Tento konfigurační soubor se samozřejmě neomezuje pouze na jmenné služby. Pro každou funkci, která je podporována balíkem NYS a vyhledává určitá data, obsahuje konfigurační soubor řádek s výčtem služeb, které se budou používat.

Správné pořadí služeb závisí na typu dat. Je málo pravděpodobné, že by mapa `services.byname` obsahovala položky, které se liší od položek uvedených v místním souboru `services`; tato mapa jich jen může obsahovat o něco více. Bylo by tedy dobré, kdyby se dotazování odehrávalo nejdříve v místních souborech a teprve v případě, že by název požadované služby nebyl nalezen, pokračovalo pomocí služby NIS. Na druhou stranu se informace o názvu hostitele mohou velmi často měnit, takže by server DNS nebo server NIS měl mít vždy nejaktuálnější informace, zatímco místní soubor `hosts` je uchováván pouze jako záložní kopie, kdyby systém DNS nebo systém NIS selhal. V tomto případě budete chtít prozkoumat místní soubor až jako poslední.

Následující příklad ukazuje způsob nastavení funkcí *gethostbyname(2)*, *gethostbyaddr(2)* a *getservbyaddr(2)* tak, aby se chovaly výše popsaným způsobem. Tyto funkce postupně zkouší uvedené služby; je-li vyhledávání úspěšné, vrátí výsledek, v opačném případě se bude zkoušet další služba.

```
# jednoduchý příklad /etc/nsswitch.conf
#
hosts:      nis dns files
services:   files nis
```

Dále následuje kompletní seznam služeb, které mohou být použity jako položky konfiguračního souboru `nsswitch.conf`. Jaké mapy, soubory a servery budou ve skutečnosti dotazovány závisí na názvu položky.

nisplus nebo *nis+* — Pro danou doménu se použije server NIS+. Umístění příslušného serveru se získá ze souboru `/etc/nis.conf`.

nis — Pro danou doménu se použije aktuální server NIS. Umístění dotazovaného serveru je nastaveno v souboru `yp.conf`, který jsme si popsali v předchozí stati. Pro položku `hosts` budou dotazovány mapy `hosts.byname` a `hosts.byaddr`.

dns — Použije se jmenný server systému DNS. Tento typ služby je užitečný pouze ve spojitosti s položkou `hosts`. Dotazované jmenné servery jsou opět převzaty ze standardního souboru `resolv.conf`.

files — Použije se místní soubor, pro položku `hosts` je to například soubor `/etc/hosts`.

dbm Informace se vyhledají v souborech typu DBM, které jsou umístěny v adresáři `/var/dbm`. Název použitého souboru vyplývá z odpovídající mapy systému NIS.

Balík NYS v současné době akceptuje v souboru `nsswitch.conf` následující položky: `hosts`, `networks`, `passwd`, `group`, `shadow`, `gshadow`, `services`, `protocols`, `rpc` a `ethers`. Časem budou pravděpodobně přidány další.

Obrázek 10.1 ukazuje o něco složitější příklad, který ukazuje novou vlastnost konfiguračního souboru `nsswitch.conf`: Klíčové slovo *[NOTFOUND=return]*, které je uvedeno v položce `hosts`, sděluje balíku NYS, aby se v případě, že nelze požadovanou položku nalézt v databázi systému NIS nebo systému DNS, vrátil zpět k dříve uvedeným volbám. To znamená, že balík NYS bude pokračovat v prohledávání místních souborů *pouze v případě*, kdy volání serveru NIS nebo serveru DNS ztroskotá z nějakého jiného důvodu. Potom budou místní soubory použity pouze při procesu zavádění nebo jako záložní kopie v případě, že server NIS není spuštěn.

```
# /etc/nsswitch.conf
#
hosts:          nis dns [NOTFOUND=return] files
networks:       nis [NOTFOUND=return] files

services:       files nis
protocols:      files nis
rpc:            files nis
```

Obrázek 10.1
Vzorový soubor `nsswitch.conf`

10.7 Použití map passwd a group

Jednou z hlavních aplikací systému NIS je synchronizace informací o uživatelích a účtech mezi všemi hostiteli v rámci příslušné domény systému NIS. V důsledku toho se používá pouze malý soubor `/etc/passwd`, ke kterému se připojují informace z map systému NIS, které obsahují data od ostatních hostitelů. Obvykle ale nestačí pouze povolit v souboru `nsswitch.conf` vyhledávání dané služby pomocí systému NIS.

Spoléháte-li se na informace s hesly šířené pomocí systému NIS, musíte se nejprve ujistit, že číselné uživatelské id libovolného uživatele, které máte umístěno ve svém lokálním souboru `passwd`, odpovídá představě serveru NIS o uživatelském id daného uživatele. Totéž budete požadovat i pro jiné účely. Příkladem může být připojení k vlastní síti svazků systému NFS ostatními hostiteli.

Pokud se některé z číselných id v souboru `/etc/passwd` nebo v souboru `/etc/group` liší od číselných id, které jsou uvedeny v mapách, musíte upravit vlastnictví všech souborů, které se vztahují k danému uživateli. Nejprve byste měli přiřadit nové hodnoty všem uid a gid v souborech `passwd` a `group`; potom byste měli vyhledat všechny soubory, které patří změněným uživatelům, a nakonec byste měli změnit vlastnictví těchto souborů. Předpokládejme, že účet **news** měl uživatelské id rovno 9 a účet **okir** měl uživatelské id rovno 103. Tato uživatelská id se změnila na jinou hodnotu; pak byste měli spustit následující příkazy:

```
# find / -uid   9 -print >/tmp/uid.9
# find / -uid 103 -print >/tmp/uid.103
# cat /tmp/uid.9   | xargs chown news
# cat /tmp/uid.103 | xargs chown okir
```

Je důležité, abyste tyto příkazy spustili s *nově* nainstalovaným souborem `passwd` a abyste si zjistili názvy všech souborů dříve, než začnete měnit vlastnictví kteréhokoliv z nich. K aktualizaci vlastnictví souborů skupiny použijte obdobnou sekvenci příkazů.

Jakmile provedete tyto změny, budou ve vašem systému souhlasit číselná uid a gid s těmi, které jsou uvedeny na všech ostatních hostitelích ve vaší doméně systému NIS. V dalším kroku přidáme do souboru `nsswitch.conf` další konfigurační řádky, které povolí vyhledávání informací o uživatelích a skupinách pomocí systému NIS:

```
# /etc/nsswitch.conf - passwd a group
passwd: nis files
group:  nis files
```

Tyto příkazy zajistí, že program `login` a všechny jeho příbuzné programy se budou při pokusu o přihlášení uživatele nejprve dotazovat map systému NIS a pokud vyhledávání pomocí systému NIS neuspěje, bude vyhledávání pokračovat v místních souborech. Ze svých místních souborů většinou odstraníte záznamy většiny uživatelů a ponecháte v nich pouze záznamy pro uživatele **root** a obecné účty, jako například účet **mail**. To proto, že některé důležité systémové úkoly mohou vyžadovat mapování uživatelských id na jména uživatelů a naopak. Například administrativní úkoly `cron` mohou spustit příkaz `su`, aby dočasně přešly na účet

news nebo může podsystém protokolu UUCP poslat zprávu o svém současném stavu na účet **uucp**. Pokud místní soubor `passwd` neobsahuje položky pro účty **news** a **uucp**, pak tyto úkoly skončí výpadem při použití systému NIS.

Zde se vynoří dvě závažné námitky: na jednu stranu bude dosud popsané nastavení fungovat pouze s přihlašovacími příkazy, které neobsahují stínová hesla. Jako příklad těchto přihlašovacích příkazů můžeme uvést programy z balíku `util-linux`. Nesnáze spojené s použitím stínových hesel společně se systémem NIS budou probrány dále. Na druhou stranu nepřistupují k souboru `passwd` pouze přihlašovací příkazy – podívejte se na příkaz `ls`, který používá většina lidí takřka permanentně. Kdykoliv provádíte nějaké dlouhé výpisy, zobrazí příkaz `ls` symbolické názvy vlastníků souboru (uživatelů a skupin); to znamená, že kdykoliv příkaz `ls` narazí na nějaké uid nebo gid, bude se muset dotazovat serveru NIS. To velmi podstatně zpomalí chod programu v případě, že je vaše místní síť ucpaná nebo, v horším případě, není server NIS ve stejné fyzické síti, takže musí datagramy procházet přes směrovač.

Zde však celé povídání zdaleka nekončí. Představte si, co se stane, když si chce uživatel změnit své heslo. Obvykle spustí příkaz `passwd`, který přijme nové heslo a aktualizuje místní soubor `passwd`. Tento postup není možný s využitím systému NIS, protože tento soubor již není lokálně dostupný. Chce-li si uživatel změnit heslo, nepomůže mu ani přihlášení k serveru NIS. Proto systém NIS poskytuje náhradu k příkazu `passwd`, a to příkaz `yppasswd`, který provádí analogickou změnu hesla, avšak při spuštěném systému NIS. Chcete-li změnit heslo na hostiteli serveru, spojí se příkaz `yppasswd` za pomoci volání RPC s démonem `yppasswdd`, který již na tomto hostiteli běží, a poskytne mu aktualizované informace o heslu. Příkaz `yppasswd` se obvykle nainstaluje místo klasického příkazu `passwd` za pomoci následující sekvence příkazů:

```
# cd /bin
# mv passwd passwd.old
# ln yppasswd passwd
```

Současně musíte na serveru nainstalovat démona `rpc.yppasswdd` a spustit ho ze skriptu `rc.inet2`. Tyto úpravy efektivně skryjí před vašimi uživateli všechny změny, které vyžaduje systém NIS.

10.8 Použití systému NIS s podporou stínových hesel

Zatím neexistuje žádná podpora systému NIS pro systémy, které používají balík pro stínová hesla. John F. Haugh, autor stínového balíku, teprve nedávno uvolnil na adrese **comp.sources.misc** verzi knihovny se stínovými funkcemi, kterou vložil do knihovny GPL, jež je sou-

částí projektu GNU. Tato knihovna již obsahovala určitou podporu pro systém NIS, ale ta stále ještě není kompletní a její soubory zatím nebyly přidány do standardní knihovny C. Na druhou stranu může uveřejnění informací ze souboru `/etc/shadow` pomocí systému NIS určitým způsobem zmařit účel použití stínového balíku.

I když funkce pro vyhledávání hesel v balíku NYS nepoužívají mapu `shadow.byname` nebo nějakou podobnou mapu, podporuje balík NYS použití souboru `/etc/shadow`. Když je zavolána příslušná implementace příkazu `getpwnam` z balíku NYS, který má vyhledat informace vztahující se k danému přihlašovacímu jménu, budou dotazovány ty prostředky, které jsou uvedeny v záznamu `passwd` v souboru `nsswitch.conf`. Služba *nis* jednoduše vyhledá název na serveru NIS v mapě `passwd.byname`. Avšak služba *files* zkontroluje, zda je přítomen soubor `/etc/shadow`, a pokud ano, pokusí se ho otevřít. Jestliže takový soubor neexistuje nebo pokud nemá uživatel práva **root**, vrátí se služba *files* k tradičnímu chování a informace o uživateli se budou vyhledávat v souboru `/etc/passwd`. Pokud však soubor `shadow` existuje a jde-li otevřít, vytáhne si z něho balík NYS uživatelovo heslo. Funkce *getpwuid* je implementována obdobným způsobem. Tímto způsobem se binární soubory zkompilované pomocí balíku NYS zřetelně vypořádají s místní instalací stínového balíku.

10.9 Použití tradičního kódu systému NIS

Používáte-li kód klienta, který je v současné době součástí standardní knihovny *libc*, bude konfigurace klienta systému NIS nepatrně odlišná. K vysílání směrem k aktivním serverům se používá démon `ypbind`, takže se informace nezískávají z konfiguračního souboru. Z toho důvodu je třeba se ujistit, že při zavádění systému dochází ke spouštění démona `ypbind`. Tento démon musí být spuštěn po nastavení domény systému NIS a po spuštění mapovače portů RPC. Použití příkazu `ypcat` pro otestování serveru by pak mělo fungovat výše popsaným způsobem.

Nedávno byl hlášen velký počet chyb, při kterých systém NIS selhal a vypsal chybovou zpsávu „`clntudp_create: RPC: portmapper failure – RPC: unable to receive`“. Tato zpráva se objevovala z důvodu nekompatibility ve způsobu, jakým démon `ypbind` sděluje funkcím v knihovně informace o vazbách. Obstarání posledních zdrojových kódů utilit systému NIS a jejich následná rekompilace by měly tento problém vyřešit.[5]

Také způsob, jakým se tradiční systém NIS rozhoduje, zda a jak má připojit informace systému NIS k informacím z místních souborů, se liší od způsobu, který používá balík NYS. Chcete-li například, aby systém NIS používal mapy s hesly, je třeba do souboru své mapy `/etc/passwd` přidat následující řádek:

5 Zdrojový kód pro balík `yp-linux` může být získán na adrese **ftp.uni-paderborn.de** v adresáři `/pub/Linux/LOCAL`.

```
+:*:0:0:::
```

Tento výraz označuje místo, kam mají funkce pro vyhledávání hesla „vložit“ mapy systému NIS. Vložíte-li podobnou řádku (bez závěrečných dvojteček) do souboru `/etc/group`, zjistíte stejnou funkci pro mapy `group.*`. Chcete-li používat mapy `hosts.*` distribuované pomocí systému NIS, změňte řádek `order` v souboru `host.conf`. Když chcete používat například systémy NIS, DNS a soubor `/etc/hosts` (v tomto pořadí), musíte změnit řádek `order` následovně:

```
order yp bind hosts
```

V současné době nepodporuje tradiční implementace systému NIS žádné další mapy.

Kapitola

11

Síťový souborový systém (NFS)

Síťový souborový systém neboli NFS je pravděpodobně nejvýznamnější síťovou službou, která využívá balík RPC. Umožňuje přístup k souborům na vzdálených hostitelích, který je uskutečňován zcela stejným způsobem jako přístup uživatele k místním souborům. Toto chování je možné díky kombinaci funkce jádra operačního systému na straně klienta (který používá vzdálený souborový systém) a serveru NFS na straně serveru (který poskytuje souborová data). Tento přístup k souborům je pro klienta zcela transparentní a funguje na nejrůznějších serverových a klientských architekturách.

Systém NFS nabízí několik výhod:

- Data, k nimž přistupují všichni uživatelé, mohou být uchovávána na centrálním hostiteli a klienti si tento adresář připojí při zavádění systému. Například můžete uchovávat všechny účty uživatelů na jediném hostiteli a všichni hostitelé ve vaší síti si z tohoto hostitele připojí adresář `home`. Je-li tento systém nainstalován společně se systémem NIS, potom se uživatelé mohou přihlásit k libovolnému systému a přitom mohou stále pracovat s jednou sadou souborů.

- Data, která zabírají velké množství diskového prostoru, mohou být uchovávána na jediném hostiteli. Například všechny soubory a programy vztahující se k balíkům LaTeX a METAFONT mohou být uloženy a spravovány na jednom místě.

- Administrativní data mohou být uchovávána na jediném hostiteli. Již není třeba používat příkaz `rcp` k nainstalování stejného souboru na 20 různých počítačích.

Na systému NFS má v Linuxu zásluhu zejména Rick Sladkey,[1] který napsal kód systému NFS pro jádro operačního systému a značnou část serveru NFS. Server NFS byl odvozen ze serveru *unfsd* (uživatelský server NFS), který původně vytvořil Mark Shand, a ze serveru *hnfs* (Harrisův server NFS), jehož autorem je Donald Becker.

Podívejme se nyní, jak systém NFS funguje: Klient může požádat o připojení adresáře ze vzdáleného hostitele ke svému místnímu adresáři naprosto stejným způsobem, jako při připojování fyzického zařízení. Avšak syntaxe používaná k zadání vzdáleného adresáře je odlišná. Chcete-li třeba připojit adresář `/home` z hostitele **vlager** k adresáři `/users` na hostiteli **vale**, měl by správce na hostiteli **vale** spustit následující příkaz:[2]

```
# mount -t nfs vlager:/home /users
```

Příkaz `mount` se pokusí spojit pomocí procedur RPC s připojovacím démonem `mountd`, který běží na hostiteli **vlager**. Server zkontroluje, zda má hostitel **vale** povoleno příslušné připojení, a pokud ano, vrátí mu souborový klíč. Tento souborový klíč bude použit ve všech následujících souborových požadavcích pod adresářem `/users`.

Při přístupu k souboru pomocí systému NFS spustí jádro systému volání procedury RPC směrem na démona `nfsd` (démon systému NFS), který se nachází na počítači serveru. Toto volání získá souborový klíč, název souboru, k němuž se má přistupovat, a jako parametry i id uživatele a skupiny, které se vztahují k danému uživateli. Tyto hodnoty slouží k určení přístupových práv k zadanému souboru. Aby se zabránilo neautorizovaným uživatelům ve čtení nebo úpravě souborů, musí být id uživatele a skupiny na obou hostitelích totožná.

Ve většině unixových implementací je systém NFS jak pro klienta, tak i pro server implementován pomocí démonů, kteří běží na úrovni jádra operačního systému a jsou spouštěni z uživatelského prostoru při zavádění systému. Na hostiteli serveru je spuštěn démon systému NFS (`nfsd`) a na hostiteli klienta je spuštěn *blokový V/V démon* (*Block I/O daemon* – `biod`). Aby se zvýšila propustnost, provádí démon `biod` asynchronní V/V komunikaci s využitím dopředného čtení a opožděného zápisu; současně je obvykle spuštěno několik démonů `nfsd`.

Implementace systému NFS v Linuxu se nepatrně liší tím, že kód klienta je pevně integrován do virtuálního systému souborů (virtual file system – VFS) v jádru operačního systému a nepotřebuje dodatečné řízení pomocí démona `biod`. Na druhou stranu běží kód serveru kompletně v uživatelském prostoru, takže je v zásadě nemožné současně spustit více kopií serveru, protože by mohlo dojít k problémům se synchronizací.

1 Ricka můžete zastihnout na adrese **jrs@world.std.com**.

2 Všimněte si, že argument `-t nfs` je možné vynechat, protože příkaz `mount` podle dvojtečky poznal, že jde o svazek systému NTFS.

Největší problém linuxového kódu systému NFS spočívá v tom, že jádro operačního systému Linux není ve verzi 1.0 schopno alokovat více paměti, než 4 KB; v důsledku toho neumí síťový kód spravovat větší datagramy, než 3500 bajtů, což je velikost datagramu po odečtení velikosti hlavičky atd. To znamená, že přenosy z a na démony systému NFS, které implicitně používají velké UDP datagramy (například datagramy o velikosti 8 KB, které používá SunOS), musí být uměle upraveny na podporovanou velikost datagramu. To může mít za určitých okolností tvrdý dopad na výkon systému.[3] Tento limit se podařilo odstranit teprve nedávno uvedením jádra operačního systému Linux verze 1.1. Kód klienta byl navíc upraven tak, aby této výhody uměl využít.

11.1 Příprava systému NFS

Před použitím systému NFS, ať už jako klienta nebo jako serveru, je třeba se ujistit, že jádro vašeho operačního systému má zkompilovanou podporu pro systém NFS. Novější jádra mají pro tento účel jednoduché rozhraní, které poskytuje informace o souborových systémech. Veškeré informace najdete v souboru `/proc/filesystems`, který zobrazíte příkazem `cat`:

```
$ cat /proc/filesystems
minix
ext2
msdos
nodev   proc
nodev   nfs
```

Pokud v tomto seznamu chybí souborový systém `nfs`, pak je nutné zkompilovat vlastní jádro operačního systému s podporou systému NFS. Konfigurace síťových voleb jádra operačního systému je probrána ve stati „Konfigurace jádra operačního systému“, která je ve 3. kapitole.

Ve starších verzích jádra operačního systému, před verzí Linuxu 1.1, je nejjednodušším způsobem, jak zjistit, zda vaše jádro má povolenou podporu systému NFS, vyzkoušet připojit nějaký souborový systém NFS. Za tím účelem můžete třeba v adresáři `/tmp` vytvořit podadresář a zkusit k němu připojit nějaký místní adresář:

```
# mkdir /tmp/test
# mount localhost:/etc /tmp/test
```

[3] Takto mi to vysvětlil pan Alan Cox: Specifikace systému NFS vyžaduje, aby server předtím, než vrátí potvrzení, uložil všechny zápisy na disk. Protože jádra balíku BSD jsou schopna zapisovat pouze data o velikosti stránky (což jsou 4 KB), budou k zápisu 4 bloků o velikosti 1 KB dat na server NFS, který používá balík BSD, použity čtyři operace zápisu, z nichž každá bude pracovat se 4 KB.

Pokud tento pokus o připojení místního adresáře selže a objeví se zpráva „`fs type nfs no supported by kernel`“, pak bude třeba vytvořit nové jádro operačního systému s povolenou podporou systému NFS. Jakékoliv další chybové zprávy jsou naprosto neškodné, protože jste zatím na svém hostiteli nenakonfigurovali démony systému NFS.

11.2 Připojení svazku systému NFS

Svazky systému NFS[4] se připojují podobně, jako běžné souborové systémy. Spustíte příkaz `mount`, který bude mít následující syntaxi:

```
# mount -t nfs nfs_volume local_dir options
```

Svazek `nfs_volume` je předán ve tvaru `remote_host:remote_dir` (`vzdálený_hostitel:vzdálený_adresář`). Protože je tento zápis jedinečný pro systémy NFS, můžete vynechat volbu `-t nfs`.

Existuje také spousta doplňkových voleb, které můžete při připojování svazku systému NFS použít v příkazu `mount`. Tyto volby mohou buď následovat na příkazové řádce za přepínačem `-o`, nebo je lze pro daný svazek uvedést v příslušné položce voleb v souboru `/etc/fstab`. V obou případech se více voleb navzájem odděluje čárkami. Volby zadané na příkazové řádce vždy potlačí volby, které jsou uvedeny v souboru `fstab`.

Vzorová položka v souboru *fstab* může vypadat takto:

```
# svazek                 připojovací bod    typ    volby
news:/usr/spool/news   /usr/spool/news    nfs    timeo=14,intr
```

Pak lze tento svazek připojit za pomoci příkazu

```
# mount news:/usr/spool/news
```

Pokud by v souboru `fstab` nebyly uvedeny žádné volby, bylo by použití příkazu `mount` na systém NFS složitější. Předpokládejme například, že si chcete připojit domovské adresáře svých uživatelů z počítače **moonshot**, který používá pro operace čtení/zápis implicitní velikost bloku 4 KB. Velikost bloku můžete snížit pomocí následujícího příkazu na 2 KB, aby vyhovovala omezení vztahujícímu se na velikosti datagramu v operačním systému Linux:

```
# mount moonshot:/home /home -o rsize=2048,wsize=2048
```

4 Nepoužíváme zde pojem souborový systém, protože systém NFS není pravým souborovým systémem.

Kompletní seznam všech korektních voleb je popsán na stránkách manuálu *nfs(5)*, jenž je součástí nástroje `mount` od Ricka Sladkeyho. Tento nástroj spolupracuje se systémem NFS (najdete ho v balíku `util-linux` od Rika Faitha). Následuje neúplný seznam voleb, které se vám mohou hodit:

rsize=n a *wsize=n* — Tyto volby určují velikosti datagramu, které budou používat klienti systému NFS u požadavků na čtení, resp. zápis. Jejich současná implicitní hodnota je, z důvodu výše popsaného omezení velikosti datagramu protokolu UDP, 1 024 bajtů.

timeo=n — Tato volba nastavuje čas (v desetinách sekundy), po který bude klient čekat na splnění jeho požadavku. Implicitní hodnota je 0,7 sekundy.

hard — Explicitně označí tento svazek jako pevně připojený svazek. Tato volba je implicitně zapnutá.

soft — Ovladač připojí svazek volně (to je opak pevného připojení).

intr — Tato volba povolí signálům přerušit volání systému NFS. Tato volba je užitečná, když potřebujete zrušit volání, například v situaci, kdy server neodpovídá.

Kromě voleb *rsize* a *wsize* všechny výše uvedené volby nastavují chování klienta pro případ, kdy server není dočasně dostupný. Tyto volby spolu souvisí následovně: kdykoliv klient pošle požadavek na server NFS, očekává, že bude operace dokončena v daném časovém limitu (ten udává volba *timeout*). Pokud v tomto intervalu nepřijde žádné potvrzení, nastane tzv. *vedlejší překročení časového limitu* (*minor timeout*) a operace se spustí znovu s dvojnásobnou hodnotou časového intervalu. Po dosažení maximálního časového intervalu 60 sekund dojde k tzv. *hlavnímu překročení časového limitu* (*major timeout*).

Hlavní překročení časového limitu implicitně způsobí, že klient vypíše na konzolu zprávu a celý proces se bude znovu opakovat, tentokrát ovšem s časovým intervalem rovným dvojnásobku časového intervalu z předchozího pokusu. Teoreticky by tento postup mohl trvat nekonečně dlouhou dobu. Svazky, které se tvrdohlavě pokoušejí opakovat operaci tak dlouho, dokud nebude server opět dostupný, se nazývají *pevně připojené* svazky. Opačný typ svazků představují tzv. *volně připojené* svazky, které při překročení hlavního časového limitu vygenerují pro daný volaný proces V/V chybu. Protože se používá opožděný zápis, který jsme si představili společně s vyrovnávací pamětí cache, nebude tato chybová zpráva předána vlastnímu procesu dříve, než bude volána funkce *write(2)*, takže program nemá nikdy jistotu, zda operace zápisu u volně připojeného svazku proběhla správně.

Zda se má daný svazek připojit pevně anebo volně není jen pouhou otázkou vkusu, ale tato volba musí záviset na typu informací, ke kterým chcete na tomto svazku přistupovat. Pokud si například pomocí systému NFS připojíte programy X, určitě nebudete chtít, aby se vaše relace chovala divně jen proto, že někdo zablokoval síť například současným spuštěním sedmi kopií programu *xv* nebo krátkodobým vytáhnutím ethernetové zástrčky. Když tento svazek připojíte pevně, budete mít jistotu, že váš počítač bude čekat tak dlouho, dokud se znovu neobnoví spojení s vaším server NFS. Na druhou stranu mohou existovat volně připojené svazky, které neobsahují kritická data, příkladem může být připojení části news pomocí systému NFS nebo připojení archívů protokolu FTP. Pokud nebude vzdálený počítač dočasně dosažitelný, nebo pokud bude vypnutý, nezablokuje volné připojení svazků vaši relaci. Je-li vaše spojení se serverem nestálé nebo prochází přes vytížený směrovač, můžete buď zvýšit počáteční hodnotu časového intervalu (pomocí volby *timeo*), nebo můžete dané svazky připojit pevně, avšak zároveň musíte povolit signály přerušující volání systému NFS, protože jinak nebudete moci přerušit zablokovaný přístup k souboru.

Démon `mountd` bude obvykle nějakým způsobem informován o tom, které adresáře jsou připojeny kterým hostitelem. Tyto informace lze zobrazit za pomoci programu `showmount`, který je taktéž součástí balíku serveru systému NFS.

11.3 Démoni systému NFS

Pokud chcete službu NFS poskytovat i ostatním hostitelům, musíte mít na svém počítači spuštěny démony `nfsd` a `mountd`. Protože se jedná o programy založené na balíku RPC, nejsou spravovány super serverem `inetd`, ale jsou spouštěny při procesu zavádění systému a sami se registrují pomocí mapovače portů. Proto se musíte ujistit, že je spouštíte až po spuštění démona `rpc.portmap`. Obvykle se do souboru `rc.inet2` přidají následující dvě řádky:

```
if [ -x /usr/sbin/rpc.mountd ]; then
        /usr/sbin/rpc.mountd; echo -n " mountd"
fi
if [ -x /usr/sbin/rpc.nfsd ]; then
        /usr/sbin/rpc.nfsd; echo -n " nfsd"
fi
```

Informace o vlastnictví, které poskytuje démon systému NFS svým klientům, obvykle obsahují pouze číselná id uživatele a skupiny. Pokud klient i server sdruží s těmito číselnými id shodné názvy uživatelů a skupin, potom říkáme, že sdílejí stejný prostor uid/gid. Příkladem tohoto sdílení může být použití systému NIS k distribuci informací ze souboru `passwd` všem hostitelům ve vaší lokální síti.

V některých případech se však tato čísla neshodují. Potom spíše aktualizujte čísla uid a gid na straně klienta tak, aby odpovídala číslům na straně serveru. K tomuto účelu můžete použít mapovacího démona `ugidd`. Použijete-li níže popsanou volbu *map_daemon*, řeknete tím démonu `nfsd`, že má prostor uid/gid nacházející se na serveru přiřadit k prostoru uid/gid, který se nachází na straně klienta. K této operaci se využije démona `ugidd`, který je spuštěný na straně klienta.

Démon `ugidd` je server založený na balíku RPC a spouští se ze souboru `rc.inet2`, stejně jako démoni `nfsd` a `mountd`.

```
if [ -x /usr/sbin/rpc.ugidd ]; then
         /usr/sbin/rpc.ugidd; echo -n " ugidd"
fi
```

11.4 Soubor exports

Zatímco výše uvedené volby se týkají konfigurace systému NFS na straně klienta, na straně serveru existuje odlišná skupina voleb, které konfigurují jeho chování vůči klientovi. Tyto volby musí být nastaveny v souboru `/etc/exports`.

Démon `mountd` implicitně nepovolí žádnému uživateli připojení adresářů z místního hostitele, což je poměrně rozumný přístup. Chcete-li jednomu nebo více hostitelům povolit připojení adresáře pomocí systému NFS, musí být tento adresář *exportován*, což znamená, že musí být uveden v souboru `exports`. Vzorový soubor `exports` by mohl vypadat následovně:

```
# soubor exports pro vlager
/home               vale(rw) vstout(rw) vlight(rw)
/usr/X386           vale(ro) vstout(ro) vlight(ro)
/usr/TeX            vale(ro) vstout(ro) vlight(ro)
/                   vale(rw,no root squash)
/home/ftp           (ro)
```

Každá řádka definuje adresář a hostitele, kteří si ho mohou připojit. Název hostitele je obvykle plně kvalifikovaným doménovým jménem, ale kromě toho může obsahovat i zástupné znaky * a *?*, které fungují stejným způsobem, jako v příkazovém interpretu `bash`. Například adrese **lab*.foo.com** vyhovuje jak adresa **lab01.foo.com**, tak i adresa **laber.foo.com**. Pokud není zadán žádný název hostitele, jako tomu bylo ve výše uvedeném příkladu s adresářem `/home/ftp`, potom si budou moci tento adresář připojit všichni uživatelé.

Při ověřování klienta pomocí souboru `exports` vyhledá démon `mountd` název hostitele klienta pomocí volání procedury *gethostbyaddr(2)*. V systému DNS vrátí tato procedura kanonický název hostitele klienta, takže je třeba se ujistit, že v souboru `exports` nepoužíváte žádné přezdívky. Bez použití systému DNS bude vrácený název odpovídat prvnímu nalezenému názvu v souboru `hosts`, který odpovídá adrese klienta.

Za názvem hostitele může následovat volitelný seznam příznaků oddělených čárkami a uzavřený do kulatých závorek. Tyto příznaky mohou nabývat následujících hodnot:

insecure — Povoluje neautorizovaný přístup z tohoto počítače.

unix-rpc — Vyžaduje ověření unixové domény pomocí balíku RPC z tohoto počítače. Tento příznak vyžaduje, aby požadavky přicházely z rezervovaného internetového portu (tj. číslo portu musí být nižší než hodnota 1 024). Tato volba je implicitně zapnutá.

secure-rpc — Vyžaduje bezpečné ověření totožnosti pomocí balíku RPC z tohoto počítače. Tato funkce zatím nebyla implementována. Viz dokumentace Secure RPC od firmy Sun.

kerberos — Aby byl umožněn přístup z tohoto počítače, je vyžadováno ověření totožnosti pomocí balíku Kerberos. Tato funkce nebyla zatím implementována. Viz dokumentace MIT, kde najdete informace o ověřovacím systému balíku Kerberos.

root_squash — Toto je bezpečnostní vlastnost, která na zadaných hostitelích zablokuje superuživatelům speciální přístupová práva. Toho se docílí přesměrováním požadavku z čísla uid 0 na straně klienta na číslo uid 65 534 (-2) na straně serveru. Toto číslo uid by mělo být přiřazeno uživateli jménem **nobody**.

no_root_squash — Nebudou se mapovat požadavky od uživatele, který má uid rovno 0. Tato volba je implicitně zapnutá.

ro — Hierarchie souborového systému se připojí v režimu pouze pro čtení. Tato volba je implicitně zapnutá.

rw — Hierarchie souborového systému se připojí v režimu pro zápis i čtení.

link_relative — Tato volba převede absolutní symbolické odkazy (to jsou takové odkazy, které začínají symbolem lomítka) na relativní odkazy. Toho se docílí připojením potřebného počtu posloupnosti znaků „../“ před vlastní název symbolického odkazu, tím se z adresáře obsahujícího daný odkaz

stane kořenový adresář na serveru. Tato volba má smysl pouze v případě, kdy je připojen celý souborový systém hostitele, v opačném případě by se mohlo stát, že některá spojení nebudou ukazovat nikam, anebo v horším případě na soubory, které by rozhodně neměly být vidět.

Tato volba je implicitně zapnutá.

link_absolute Zanechá všechny symbolické odkazy v původním stavu (toto je normální chování serverů NFS, které jsou dodávány firmou Sun).

map_identity Volba *map_identity* říká serveru, aby předpokládal, že klient používá stejná čísla uid a gid jako server. Tato volba je implicitně zapnutá.

map_daemon Tato volba sdělí serveru NFS, aby předpokládal, že klient a server nesdílí společný prostor s čísly uid/gid. Potom démon `nfsd` vytvoří seznam mapující čísla id mezi klientem a serverem. Ty získá tak, že se bude na straně klienta dotazovat démona `ugidd`.

Chyba zjištěná při analýze souboru `exports` bude zapsána do souboru `syslog` s úrovní *notice* (*oznámení*). Tato chyba se zapíše vždy, když je spuštěn démon `nfsd` nebo `mountd`.

Pamatujte, že názvy hostitelů jsou získány z IP-adresy klienta pomocí zpětného mapování, takže je nutné mít správně nakonfigurovaný resolver. Pokud používáte službu BIND a zakládáte si na bezpečnosti, měli byste v souboru `host.conf` povolit kontrolu spoofingu.

11.5 Automatické připojování (automounter) v operačním systému Linux

V některých případech není hospodárné připojovat všechny svazky systému NFS, ke kterým by mohli chtít uživatelé přistupovat; buď z důvodu pouhého množství připojovaných svazků, nebo z důvodu časové prodlevy, kterou by si tento proces vyžádal při zavádění systému. Schůdnou alternativou řešení tohoto problému je tzv. *démon automatického připojování*. Jedná se o démona, který podle potřeby automaticky a zcela transparentně připojuje libovolné svazky systému NFS a taktéž je automaticky odpojuje, pokud nebyly po určitou dobu používány. Další šikovnou vlastností, kterou disponuje démon automatického připojování, je možnost připojení určitého svazku z alternativních míst. Například kopie programů X a podpůrných souborů je možné uchovávat pouze na dvou nebo třech hostitelích a všichni ostatní hostitelé si tyto programy a podpůrné soubory připojí pomocí systému NFS. Když k tomuto účelu použijete démona automatického připojování, můžete všechny tři hostitele namapovat do adresáře `/usr/X386`; démon automatického připojování se potom bude zkoušet připojit ke všem třem hostitelům, dokud se k některému nepřipojí.

Běžně používaný démon automatického připojování se v Linuxu nazývá `amd`. Původně ho napsal Jan-Simonem Pendry a na platformu Linuxu ho portoval Rick Sladkey. Aktuální verze tohoto démona je `amd-5.3`.

Vysvětlení démona `amd` přesahuje rozsah této kapitoly; jako dobrý manuál by vám však mohl posloužit jeho zdrojový kód; tento zdrojový kód obsahuje soubor s textovými informacemi, které jsou velmi podrobné.

Kapitola

12

Správa protokolu Taylor UUCP

12.1 Historie

Protokol UUCP byl navržen koncem sedmdesátých let panem Mikem Leskem ze společnosti AT&T Bell Laboratories. Tento protokol měl poskytovat jednoduché připojení k síti pomocí veřejných telefonních linek. Protože většina lidí, kteří chtějí mít na svém domácím počítači elektronickou poštu a usenetové news, stále používá ke komunikaci modemy, zůstává protokol UUCP i nadále velice populární. I přes vysoký počet implementací tohoto protokolu, které lze provozovat na širokém spektru hardwarových platforem a operačních systémů, jsou tyto implementace navzájem vysoce kompatibilní.

Nicméně jako u většiny softwaru, u něhož se až v průběhu let vyvinul určitý „standard“, neexistuje ani u protokolu UUCP taková implementace, která by se nazývala jednoduše UUCP. Protokol UUCP prošel od své první verze, jež byla uvedena v roce 1976, neustálým evolučním vývojem. V současné době existují dva hlavní druhy tohoto protokolu, které se navzájem značně odlišují jak v podporovaném hardwaru, tak i ve své vlastní konfiguraci. Z těchto dvou druhů protokolu UUCP se vyvinula spousta různých implementací, které se od svých předků liší jen minimálně.

Jeden druh se nazývá „protokol UUCP verze 2“ a jeho vznik je datován do roku 1977, kdy Mike Lesk, David A. Novitz a Greg Chesson vytvořili novou implementaci protokolu UUCP. I když je tato implementace poměrně stará, stále se hojně používá. Novější implementace verze 2 poskytuje spoustu vymožeností obsažených v novějších druzích protokolu UUCP.

Druhý typ protokolu byl vytvořen v roce 1983 a běžně se označuje názvy BNU (Basic Networking Utilities – základní síťové utility) nebo HoneyDanBer UUCP, zkráceně HDB. Tento název je odvozen ze jmen autorů, P. Honeymana, D. A. Novitze a B. E. Redmana. Protokol HDB byl vymyšlen proto, aby odstranil některé nedostatky protokolu UUCP verze 2. Byly

například přidány nové transportní protokoly a dočasný odkládací adresář byl rozdělen tak, že v současnosti existuje jeden adresář pro každý systém, se kterým komunikujete na bázi protokolu UUCP.

V současnosti se společně s operačním systémem Linux dodává implementace protokolu UUCP pod názvem Taylor UUCP verze 1.04.[1] Z této verze vychází i tato kapitola. Protokol Taylor UUCP verze 1.04 byl dán do oběhu v únoru roku 1993 a bývá zkompilován tak, aby kromě tradičních konfiguračních souborů ovládal i konfigurační soubory nové generace – tzv. „Taylorovy“ konfigurační soubory.

Nedávno byla uvolněna verze 1.05 a již brzy se tato verze stane součástí většiny distribucí Linuxu. Odlišnosti mezi těmito verzemi se většinou týkají rysů, které nikdy nebudete používat, takže na základě informací uvedených v této knize budete moci nakonfigurovat i protokol Taylor UUCP verze 1.05.

Ve většině distribucí Linuxu je obvykle protokol UUCP zkompilován tak, aby byl buď kompatibilní s utilitami BNU nebo kompatibilní s Taylorovým konfiguračním schématem, případně s oběma verzemi. Protože je Taylorovo konfigurační schéma mnohem flexibilnější a pravděpodobně i snáze pochopitelné, než často nesrozumitelné konfigurační soubory utilit BNU, budu v této kapitole popisovat Taylorovo konfigurační schéma.

Účelem této kapitoly není poskytnout vyčerpávající popis všech voleb příkazové řádky pro všechny existující příkazy protokolu UUCP a popis funkce všech těchto příkazů, nýbrž úvod do problematiky nastavení funkčního uzlu UUCP. Doufejme, že první stať vám poskytne zevrubný úvod do problematiky implementace vzdáleného spouštění a přenosu souborů pomocí protokolu UUCP. Nejste-li úplným nováčkem v oblasti práce s protokolem UUCP, můžete tuto kapitolu přeskočit a přejít na stať zabývající se konfiguračními soubory protokolu UUCP, která vysvětluje použití různých souborů pro nastavení protokolu UUCP.

Budeme ale předpokládat, že znáte programy z balíku UUCP. Konkrétně jsou to programy *uucp* a *uux*. Popis těchto příkazů naleznete na on-line manuálových stránkách.

Kromě veřejně dostupných programů *uux* a *uucp* obsahuje balík UUCP také množství příkazů, které se používají pouze k administrativním účelům. Používají se k monitorování UUCP dopravy ve vašem uzlu, k odstraňování starých souborů se záznamy nebo k sestavování statistik. Ani jeden z těchto příkazů zde nebudeme rozebírat, protože nejsou pro hlavní činnost protokolu UUCP důležité. Kromě toho je jejich dokumentace na velmi slušné úrovni a člověk jim snadno porozumí. Existuje však ještě třetí kategorie, která představuje skutečného „zá-

1 Tuto implementaci napsal v roce 1993 Ian Taylor, který vlastní i autorská práva.

vodního koně“ protokolu UUCP. Tyto programy se nazývají `uucico` (kde zkratka cico znamená copy-in copy-out – kopie dovnitř kopie ven) a `uuxqt`, který spouští úkoly přijaté ze vzdálených systémů.

12.1.1 Další informace o protokolu UUCP

Ti, co nenajdou vše potřebné v této kapitole, by si měli přečíst dokumentaci, která je součástí balíku UUCP. Tato dokumentace se skládá z několika textových souborů popisujících nastavení při použití Taylorova konfiguračního schématu. Textové informace mohou být pomocí příkazů `tex` a `makeinfo` převedeny do informačních souborů formátu DVI, resp. GNUinfo.

Pokud chcete používat konfigurační soubory kompatibilní s utilitami BNU nebo dokonce (a z toho mě mrazí v zádech) konfigurační soubory verze 2, mohla by vám pomoci kvalitní kniha „Managing UUCP and Usenet“ ([Oreilly89]). Považuji ji za velmi užitečnou. Dalším velmi dobrým zdrojem informací o linuxovém protokolu UUCP je informační dokument UUCP-HOWTO od Vince Skahana.

Dále existuje diskusní skupina zabývající se protokolem UUCP, která se nazývá **comp.mail.uucp**. Máte-li specifické dotazy ohledně Taylorova konfiguračního schématu, bude lepší, když se na ně zeptáte přímo v této diskusní skupině, než ve skupině **comp.os.linux**.

12.2 Úvod

12.2.1 Přehled protokolu UUCP - přenos souborů a vzdálené spouštění

Pro pochopení protokolu UUCP je důležité porozumět konceptu *úkolů*. Každý přenos iniciovaný uživatelem pomocí příkazů `uucp` nebo `uux` se nazývá *úkol*. Ten se skládá z *příkazu*, který má být spuštěn na vzdáleném systému, a ze skupiny *souborů*, které mají být mezi propojenými systémy přeneseny. Jednu z těchto částí lze vynechat.

Jako příklad budeme předpokládat, že jste na svém hostiteli spustili následující příkaz, který sdělí protokolu UUCP, aby překopíroval soubor `netguide.ps` na hostitele **pablo** a aby spustil příkaz `lpr`, který tento soubor vytiskne.

```
$ uux -r pablo!lpr !netguide.ps
```

Protokol UUCP se většinou hned nespojí se vzdáleným systémem, aby provedl patřičný úkol (okamžitě to lze provést pomocí programu `kermit`). Místo toho dočasně uloží popis daného úkolu. Tento proces se nazývá *dočasné odkládání*. Strom s adresáři, ve kterém jsou uloženy jednotlivé úkoly, se nazývá *dočasný adresář* a je zpravidla umístěn v adresáři `/var/spo-`

ol/uucp. V našem příkladu by měly informace o úkolu obsahovat vzdálený příkaz (lpr), který se má spustit, jméno uživatele, který o toto spuštění žádá, a několik dalších položek. Kromě popisů jednotlivých úkolů musí protokol UUCP ukládat vstupní soubor, konkrétně soubor netguide.ps.

Přesné umístění a názvy dočasných odkládacích souborů se mohou lišit v závislosti na nastavení některých voleb při sestavování. Protokol UUCP kompatibilní s protokolem HDB zpravidla dočasně ukládá soubory v adresáři s názvem /var/spool/uucp/*site*, kde *site* je název vzdáleného systému. Když je protokol UUCP zkompilován s podporou Taylorova konfiguračního schématu, vytvoří protokol UUCP pro různé typy dočasně odkládaných souborů podadresáře v dočasném adresáři konkrétního systému.

Protokol UUCP se pak v pravidelných intervalech spojuje se vzdáleným systémem. Když se uskuteční spojení se vzdáleným systémem, přenese protokol UUCP soubory popisující daný úkol a všechny vstupní soubory. Příchozí úkoly nebudou spuštěny okamžitě, ale až po skončení spojení. To provede příkaz uuxqt, který se rovněž stará o doručení úkolů, které jsou určeny pro jiný systém.

Aby protokol UUCP rozlišil důležité a méně důležité úkoly, přiřazuje každému úkolu *prioritu*. Priorita je definována jediným znakem v rozmezí od 0 do 9, od A do Z a od a do z. Hodnota 0 odpovídá nejvyšší prioritě, hodnota z odpovídá prioritě nejnižší. Pošta je obvykle dočasně ukládána s prioritou B nebo C, zatímco news jsou ukládány s prioritou N. Úkoly s vyšší prioritou jsou přenášeny dříve. Prioritu můžete přiřadit příkazům uucp a uux pomocí parametru -g.

V určitých časových intervalech můžete také zabránit přenášení úkolů, které mají nižší prioritu, než zadáte. Tato vlastnost se také označuje jako tzv. *maximální odkládací priorita* povolená při komunikaci a implicitně je nastavena na hodnotu z. Všimněte si této terminologické dvojsmyslnosti: soubor je přenesen pouze v případě, že má prioritu *vyšší* nebo *shodnou* s maximální odkládací prioritou.

12.2.2 Vnitřní funkce programu uucico

Abyste pochopili, proč potřebuje program uucico znát určitá data, bude vhodné uvést rychlý popis toho, jakým způsobem se tento program ve skutečnosti spojuje se vzdáleným systémem.

Když na příkazové řádce spustíte příkaz uucico -s system, musí se příkaz uucico nejprve fyzicky připojit. Provedené akce budou záviset na typu navazovaného spojení – například při použití telefonní linky musí příkaz uucico nejprve najít modem a vytočit číslo. U spojení pomocí protokolu TCP musí nejprve zavolat funkci *gethostbyname(3)*, aby převedl název hostitele na síťovou adresu, poté musí zjistit port, který se má otevřít, a přiřadit adresu odpovídajícímu socketu.

Po navázání spojení je třeba provést ověření totožnosti. To se zpravidla skládá z výzvy vzdálenému systému, aby zadal přihlašovací jméno a volitelně i heslo. Tento proces se obecně nazývá *přihlašovací komunikace*. Procedura ověření totožnosti se buď provede pomocí běžného balíku `getty/login`, nebo – u socketů protokolu TCP – vlastním programem `uucico`. Pakliže je ověření totožnosti úspěšné, spustí se na vzdáleném konci program `uucico`. Místní kopie programu `uucico`, která byla původcem spojení, se označuje jako *řídicí* a vzdálená kopie se označuje jako *řízená*.

Dále následuje *inicializační fáze*: řídicí program pošle svůj název hostitele společně s několika příznaky. Řízený program ověří povolení k přihlášení pro zadaný název hostitele, pošle a přijme soubory atd. Příznaky popisují (kromě jiných vlastností) maximální prioritu dočasně ukládaných souborů, které se budou přenášet. Pokud je povolena sekvenční kontrola hovoru, uskuteční se v této fázi i ověření počtu hovorů neboli *sekvenčního počtu hovorů*. Tato volba způsobí, že si budou oba systémy hlídat počet úspěšných spojení a ty pak porovnají. Pokud si počty úspěšných spojení neodpovídají, pak inicializační fáze neuspěje. Tato vlastnost je užitečná při ochraně vašeho systému před potenciálními podvodníky.

A nakonec se oba programy `uucico` pokusí dohodnout na společném *přenosovém protokolu*. Tento protokol řídí způsob, jakým jsou přenášena data, ověřuje soudržnost dat a dojde-li k chybě, pošle data znovu. Více přenosových protokolů je zapotřebí z důvodu odlišných podporovaných typů spojení. Například telefonní linky vyžadují „bezpečný“ protokol, který je z hlediska výskytu chyb značně pesimistický, zatímco přenos pomocí protokolu TCP je již ze své podstaty spolehlivý, a proto může používat efektivnější protokol, který vynechává většinu dodatečných detekcí chyb.

Po skončení inicializační fáze začne skutečná přenosová fáze. Oba konce zapnou ovladač vybraného protokolu. Ovladače mohou eventuálně provést inicializační sekvenci, která závisí na typu vybraného protokolu.

Nejprve pošle řídicí program vzdálenému systému všechny soubory, které čekají ve frontě a jejichž priorita dočasného uložení je dostatečně veliká. Když tento přenos dokončí, bude řízený program informován o ukončení přenosu a o tom, že eventuálně může zavěsit. Řízený program nyní může buď souhlasit se zavěšením, nebo může převzít řízení komunikace. Takže de facto dojde k výměně rolí: nyní se program na vzdáleném systému stane řídicím a program na místním hostiteli se stane řízeným. Nový řídicí program nyní pošle své soubory. Po skončení tohoto přenosu souborů si oba programy `uucico` vymění zavěšující řetězce a spojení se přeruší.

Při popisu celého procesu nebudeme zabíhat do větších detailů: chcete-li více informací, nahlédněte buď do zdrojového kódu programu, anebo do nějaké kvalitní knihy, která se zabývá problematikou protokolu UUCP. Kromě toho se někde po Internetu pohybuje poměrně starý

článek od Davida A. Novitze, ve kterém najdete detailní popis protokolu UUCP. Informační bulletin FAQ k protokolu Taylor UUCP také obsahuje některé podrobnosti ohledně způsobu implementace protokolu UUCP. Pravidelně je posílán na adresu **comp.mail.uucp**.

12.2.3 Volby příkazové řádky programu uucico

Tato stať popisuje nejdůležitější volby příkazové řádky programu *uucico*. Kompletní seznam těchto voleb získáte na manuálových stránkách *uucico(1)*.

`-s system` Zavolá uvedený systém, pokud to není v době, kdy je to zakázáno. V konfiguračních souborech lze určit časové rozmezí, ve kterém je možné se s daným systémem spojit.

`-S system` Bez jakýchkoliv podmínek zavolá uvedený systém `system`.

`-r1` Spustí program `uucico` v řídícím režimu. Tento režim je implicitní, použijete-li parametry `-s` nebo `-S`. Samostatná volba `-r1` způsobí, že se program `uucico` pokusí zavolat všechny systémy uvedené v souboru *sys*, pokud toto volání není zakázáno časovým rozmezím pro volání nebo dobou pro opětovný pokus.

`-r0` Spustí program `uucico` v řízeném režimu. Tento režim je implicitní, pokud nejsou zadány parametry `-s` nebo `-S`. V řízeném režimu se předpokládá, že standardní vstup/výstup bude připojen buď na sériový port, nebo se použije port pro protokol TCP zadaný pomocí volby `-p`.

`-x type`, `-X type` Zapne ladění zadaného typu. Pomocí seznamu odděleného čárkami může být zadáno několik typů ladění. Platné jsou následující typy: *abnormal*, *chat*, *handshake*, *uucp-proto*, *proto*, *port*, *config*, *spooldir*, *execute*, *incoming*, *outgoing*. Použijete-li klíčové slovo *all*, zapnou se všechny volby. Z důvodu kompatibility s ostatními implementacemi protokolu UUCP může být zadáno místo názvu číslo, které zapíná ladění prvních `n` položek z výše uvedeného seznamu.

Ladící informace budou zapsány do souboru `Debug`, který se nachází v podadresáři `/var/spool/uucp`.

12.3 Konfigurační soubory protokolu UUCP

Na rozdíl od jednoduchých programů pro přenos souborů byl protokol UUCP navržen tak, aby se uměl automaticky postarat o všechny přenosy. Když se vám povede tento protokol správně nastavit, nebude nutný každodenní zásah ze strany správců systému. Požadované konfigurační informace jsou uchovávány v několika *konfiguračních souborech*, které jsou umístěny v adresáři `/usr/lib/uucp`. Většina z těchto souborů je používána pouze při volání směrem z vašeho systému.

12.3.1 Jemný úvod do protokolu Taylor UUCP

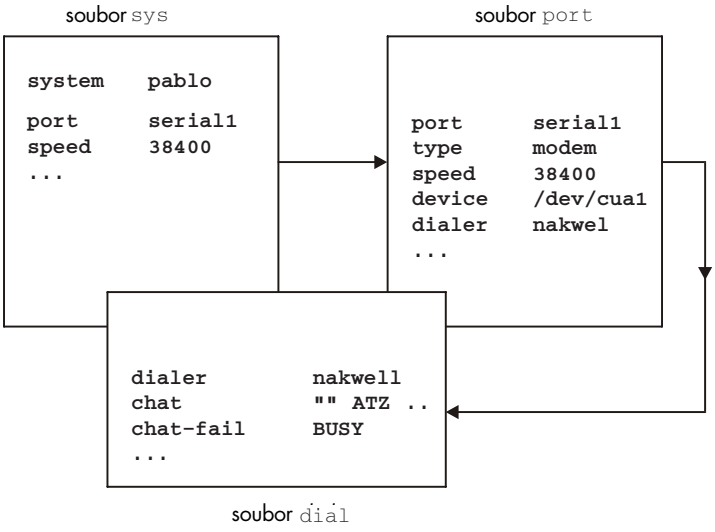
Lze říci, že konfigurace protokolu UUCP je poměrně těžká na pochopení. Jde skutečně o nepříjemnou záležitost a stručný formát konfiguračních souborů vám to rozhodně neulehčí (i když Taylorův formát je v porovnání se staršími formáty protokolů HDB nebo verze 2 poměrně čitelný).

Abyste poznali, jak spolu všechny tyto konfigurační soubory souvisí, představíme vám alespoň nejdůležitější z nich a podíváme se na vzorové položky těchto souborů. Teď se nebudeme zabývat detaily; přesnější vysvětlení najdete v následujících samostatných statích. Chcete-li, aby váš počítač podporoval protokol UUCP, bude nejlepší začít s několika vzorovými soubory, které budete postupně upravovat. Jako vzorové soubory můžete použít níže uvedené příklady nebo soubory, které jsou součástí vaší oblíbené distribuce operačního systému Linux.

Všechny soubory popisované v této stati jsou uloženy v adresáři `/usr/lib/uucp` nebo v některém z jeho podadresářů. Některé distribuce Linuxu obsahují binární soubory protokolu UUCP, které mají zkompilovanou podporu jak pro konfiguraci podle standardu HDB, tak i pro Taylorovu konfiguraci, a pro každý druh konfiguračních souborů používají jiné podadresáře. V adresáři `/usr/lib/uucp` se obvykle nachází soubor `README`.

Aby protokol UUCP správně fungoval, musí být vlastníkem jeho souborů uživatel **uucp**. Některé z těchto souborů obsahují hesla a telefonní čísla, a proto by měly mít nastavena přístupová práva 600.[2]

[2] Všimněte si, ačkoliv většina příkazů protokolu UUCP musí mít nastaveno uid na hodnotu **uucp**, musíte se ujistit, že program `uuchk` nemá přiděleno toto uid. V opačném případě by si mohli uživatelé prohlížet hesla, i když by tato hesla měla nastavena přístupová práva 600.



Obrázek 12.1
Vztahy mezi jednotlivými konfiguračními soubory protokolu Taylor UUC.

Ústředním souborem protokolu UUCP je soubor `/usr/lib/uucp/config`, který se používá pro nastavení obecných parametrů. Nejdůležitějším z těchto parametrů (a momentálně i jediným parametrem) je název vašeho hostitele protokolu UUCP. Ve společnosti Virtual Brewery používají jako bránu protokolu UUCP hostitele **vstout**:

```
# /usr/lib/uucp/config - hlavní konfigurační soubor UUCP
hostname          vstout
```

Dalším důležitým konfiguračním souborem je soubor `sys`. V něm jsou obsaženy veškeré informace týkající se systémů, s nimiž jste propojeni. Tyto informace obsahují název systému a informace o vlastním spojení, například telefonní číslo používané při modemovém spojení. Typický záznam v souboru *sys* pro systém s názvem **pablo** připojený pomocí modemu vypadá asi takto:

```
# /usr/lib/uucp/sys - sousední počítače UUCP
#
system                pablo
time                  Any
phone                 123-456
port                  serial1
speed                 38400
chat                  ogin: vstout ssword: lorca
```

Pole *port* označuje používaný port a pole *time* určuje časy, ve kterých se lze s daným systémem spojit. Pole *chat* popisuje přihlašovací komunikační skripty – souslednost řetězců, které si musí systémy navzájem vyměnit dříve, než se může program `uucico` přihlásit k hostiteli **pablo**. Ke komunikačním skriptům se vrátíme později. Příkaz *port* neoznačuje speciální soubor zařízení, jako je například `/dev/cua1`, ale označuje záznam v souboru *port*. Název portu může mít libovolný název, ale musí odkazovat na korektní záznam v souboru *port*.

Soubor `port` obsahuje informace týkající se vlastního spojení. U modemových spojení tento soubor popisuje speciální soubor používaného zařízení, rozsah podporovaných rychlostí a typ volacího zařízení, které je připojeno k danému portu. Níže uvedený záznam popisuje soubor zařízení `/dev/cua1` (což odpovídá sériovému portu COM 2), ke kterému je připojen modem NakWell, jenž je schopný pracovat s rychlostmi až do 38 400 bps. Název záznamu byl zvolen tak, aby odpovídal názvu portu, který je uveden v souboru `sys`.

```
# /usr/lib/uucp/port - porty UUCP
# /dev/cua1 (COM2)
port                serial1
type                modem
device              /dev/cua1
speed               38400
dialer              nakwell
```

Informace vztahující se k vytáčecímu zařízení jsou uchovávány v dalším speciálním souboru, jehož název jistě uhodnete – `dial`. Pro každý typ vytáčeného zařízení obsahuje tento soubor sekvenci příkazů, které musí být spuštěny, aby se pomocí zadaného čísla vytočil vzdálený systém. Této sekvenci příkazů se také říká komunikační skript. Například záznam pro výše uvedený modem NakWell by vypadal nějak takto:

```
# /usr/lib/uucp/dial - informace o vytáčecích zařízeních
# modemy NakWell
dialer          nakwell
chat            "" ATZ OK ATDT\T CONNECT
```

Řádek začínající polem *chat* definuje komunikační řetězec modemu skládající se z posloupnosti příkazů poslaných na modem a přijatých z modemu. Tyto příkazy modem inicializují a vytáčejí požadované číslo. Sekvenci znaků „\T“ nahradí program `uucico` zadaným telefonním číslem.

Abychom vám mohli poskytnout hrubé schéma způsobu práce programu `uucico` s konfiguračními soubory, budeme předpokládat, že jste spustili na příkazovém řádku následující příkaz:

```
$ uucico -s pablo
```

Program `uucico` nejprve vyhledá hostitele **pablo** v souboru `sys`. Ze záznamu hostitele **pablo**, který je umístěn v souboru `sys`, program `uucico` zjistí, že ke spojení by měl použít port `serial1`. Soubor `port` sdělí programu `uucico`, že požadovaný port je modemový port a že je k tomuto portu připojen modem NakWell.

Nyní vyhledá program `uucico` v souboru `dial` záznam, který popisuje modem NakWell, a když takový záznam nalezne, otevře sériový port `/dev/cua1` a spustí vytáčecí komunikační skript. To znamená, že pošle modemu příkaz „`ATZ`“ a bude čekat na odpověď „`OK`“ atd. Jakmile najde řetězec „\T“, nahradí ho skutečným telefonním číslem (123-456), které získá ze souboru `sys`.

Poté, co modem vrátí odpověď „`CONNECT`“, se uskuteční vlastní spojení a komunikační skript modemu skončí. Nyní se program `uucico` vrátí k souboru `sys` a spustí přihlašovací komunikační skript. V našem příkladu bude čekat na výzvu „`login:`“, potom pošle jméno uživatele (`vstout`), vyčká na výzvu „`password:`“ a následně pošle heslo „`lorca`“.

Po skončení ověřování totožnosti se předpokládá, že vzdálený konec spustí svůj vlastní program `uucico`. Potom oba tyto programy vstoupí do inicializační fáze, která byla popsána v předcházející stati.

Způsob, jakým na sobě závisí jednotlivé konfigurační soubory, je ukázán na obrázku 12.1.

12.3.2 Co potřebuje znát protokol UUCP

Dříve, než začnete sestavovat konfigurační soubory protokolu UUCP, musíte získat některé informace, které potřebuje protokol UUCP vědět.

Nejprve musíte zjistit, ke kterému sériovému zařízení je připojen váš modem. Porty COM 1 až COM 4 (v DOSu) se obvykle mapují na speciální soubory zařízení `/dev/cua0` až `/dev/cua3`. Většina distribucí, příkladem budiž distribuce Slackware, vytvoří soubor `/dev/modem`, který je symbolickým odkazem na patřičný soubor zařízení `cua*` . Tyto distribuce pak nakonfigurují programy `kermit`, `seyon` atd. tak, aby používaly obecný soubor `modem`. V tomto případě byste měli ve své konfiguraci protokolu UUCP také používat soubor `/dev/modem`.

Soubor `modem` byste měli používat z toho důvodu, že všechny programy obsluhující modem používají k signalizaci používání daného sériového portu tzv. zamykací soubory. Názvy těchto zamykacích souborů se tvoří kombinací řetězce `LCK.`. s názvem souboru příslušného zařízení, například `LCK..cua1`. Pokud by programy používaly pro stejné zařízení odlišné názvy, nebudou schopny zjistit vzájemné zamykací soubory. V důsledku toho dojde k ukončení obou relací, pokud jsou tyto relace spuštěny současně. To však není příliš pravděpodobné, plánujete-li své hovory UUCP s využitím tabulky `crontab`.

Podrobnosti týkající se nastavení sériových portů získáte v kapitole 4.

Nyní se podíváme, jakou rychlostí komunikuje váš modem s operačním systémem Linux. Tuto rychlost musíte nastavit na maximální očekávanou efektivní přenosovou rychlost. Efektivní přenosová rychlost může být mnohem vyšší, než skutečná fyzická rychlost podporovaná vaším modemem. Například mnoho modemů posílá a přijímá data rychlostí 2 400 bps (bitů za sekundu). Při použití kompresních protokolů, jako je V.42bis, se může skutečná přenosová rychlost vyšplhat až na 9 600 bps.[5]

Má-li být protokol UUCP k něčemu užitečný, budete samozřejmě potřebovat telefonní číslo volaného systému. Taktéž budete potřebovat korektní přihlašovací číslo id a volitelně i heslo.[3]

Dále musíte *přesně* vědět, jak se lze do daného systému přihlásit. Musíte třeba předtím, než se objeví výzva k přihlášení, stisknout klávesu BREAK? Zobrazuje výzva řetězec `login:` nebo `user:`? Tyto údaje jsou důležité pro sestavení *komunikačního skriptu*, který si můžeme představit jako recept, jenž sděluje programu `uucico`, jak se má přihlásit. Pokud tyto údaje neznáte nebo pokud selže i obvyklý komunikační skript, pokuste se dovolat na daný systém pomocí terminálového programu, jako je `kermit` nebo `minicom`, a zapište si přesně ty úkony, které musíte provést.

12.3.3 Pojmenování systému

Stejně jako u sítí na bázi protokolu TCP/IP musí mít váš hostitel i v sítích na bázi protokolu UUCP nějaký název. Pokud hodláte používat protokol UUCP pouze na přenos souborů do nebo ze systémů, se kterými se přímo spojujete, nebo hodláte-li používat tento protokol pouze pro přenos souborů v místní síti, nemusí tento název splňovat žádné standardy.[4]

Používáte-li však protokol UUCP pro příjem pošty nebo news, měli byste přemýšlet o tom, zda by nebylo lepší nechat si zaregistrovat váš název pomocí projektu mapování názvů pro protokol UUCP (UUCP Mapping project). Projekt mapování názvů pro protokol UUCP je popsán v kapitole 13. Dokonce i když jste členem domény, měli byste zvážit použití oficiálního názvu pro protokol UUCP.

Lidé si často zvolí svůj název pro protokol UUCP tak, aby odpovídal první části plně kvalifikovaného názvu domény. Předpokládejme, že adresa domény vašeho systému je **swim.twobirds.com**, potom by název vašeho hostitele pro protokol UUCP měl být **swim**. Pamatujte

3 Pokud si pouze chcete vyzkoušet protokol UUCP, sežeňte si číslo archivního systému, který se nachází v blízkosti vašeho počítače. Poznamenejte si přihlášení do systému a heslo – vzhledem k tomu, že se jedná o veřejné systémy, je možné provádět i anonymní stahování souborů. Jméno uživatele a heslo představuje obvykle kombinace podobných řetězců, jako je **uucp/uucp** nebo **nuucp/uucp**.

4 Jediným omezením je maximální délka názvu 7 znaků. Nesmíme si to plést s hostiteli, kteří mají souborové systémy podporující pouze velmi malý počet znaků v názvu souboru.

5 Pozn. V současnosti jsou rychlosti podstatně vyšší.

na to, že systémy UUCP se navzájem nepoznávají na základě svého názvu. Samozřejmě, že můžete používat název, který nemá nic společného s plně kvalifikovaným názvem domény.

Ujistěte se však, že v poštovních adresách nepoužíváte nekvalifikovaný název systému, pokud ho nemáte zaregistrován jako oficiální název pro protokol UUCP.[6] Pošta poslaná na neregistrovaného hostitele protokolu UUCP by v lepším případě zmizela v nějaké černé díře. Použijete-li název, který již používá nějaký jiný systém, bude pošta směrována na tento systém a způsobí poštmistrům tohoto systému značné problémy.

Balík UUCP implicitně používá název systému nastavený pomocí příkazu `hostname`. Tento název je obecně nastaven ve skriptu `/etc/rc.local`. Pokud se váš název pro protokol UUCP liší od názvu, který jste přiřadili vašemu hostiteli, musíte použít v souboru `config` volbu *hostname*, která sdělí programu `uucico` název pro protokol UUCP. Tento postup bude popsán dále.

12.3.4 Taylorovy konfigurační soubory

Nyní se vrátíme zpět ke konfiguračním souborům. Protokol Taylor UUCP získává potřebné informace z následujících souborů:

`config` Toto je hlavní konfigurační soubor. Zde můžete zadat název vašeho systému pro protokol UUCP.

`sys` Tento soubor popisuje všechny systémy, které znáte. Pro každý systém je v tomto souboru uveden název systému, časy, ve kterých se lze s tímto systémem spojit, volané číslo (pokud nějaké existuje), typ používaného zařízení a způsob, jakým se lze k tomuto systému přihlásit.

`port` Obsahuje položky popisující každý dostupný port, u kterého jsou uvedeny podporovaná rychlost linky a typ používaného vytáčecího zařízení.

`dial` Popisuje vytáčecí zařízení, které se používá pro spojení po telefonní lince.

`dialcode` Obsahuje expanze pro jednotlivé symbolické vytáčecí kódy.

`call` Obsahuje přihlašovací jméno a heslo, které se použije při volání daného systému. Používá se jen zřídka.

`passwd` Obsahuje přihlašovací jména a hesla, která mohou systémy používat při přihlašování. Tento soubor se používá pouze v případě, že si program `uucico` sám provádí ověření hesla.

6 Projekt mapování názvů pro protokol UUCP registruje po celém světě všechny názvy hostitelů pro protokol UUCP a zajišťuje jejich jedinečnost. Chcete-li si zaregistrovat svůj název pro protokol UUCP, zeptejte se správce systému, který vám poskytuje poštu.

Taylorovy konfigurační soubory se zpravidla skládají z řádků obsahujících páry hodnot klíčových slov. Symbol (#) značí komentář, který sahá až do konce příslušného řádku. Chcete-li použít vlastní symbol (#), zadejte ho společně s nahrazujícím symbolem zpětného lomítka.

Pomocí těchto konfiguračních souborů můžete ovládat poměrně velké množství voleb. Nebudeme zde probírat všechny parametry, proto si probereme pouze nejdůležitější z nich. Tyto volby by vám měly pomoci nastavit modemové spojení pomocí protokolu UUCP. Další stati budou popisovat nutné úpravy, které je třeba učinit, budete-li chtít používat protokol UUCP v síti na bázi protokolu TCP/IP nebo společně s přímou sériovou linkou. Kompletní popis je uveden v Texinfo dokumentech, které doprovázejí zdrojový kód protokolu Taylor UUCP.

Pokud si myslíte, že máte svůj systém s protokolem UUCP kompletně nakonfigurován, můžete si svoji konfiguraci zkontrolovat pomocí nástroje `uuchk` (ten najdete v adresáři `/usr/lib/uucp`). Tento nástroj přečte vaše konfigurační soubory a vytiskne podrobnou zprávu obsahující konfigurační hodnoty, které jste použili u každého systému.

12.3.5 Všeobecné konfigurační volby – soubor config

Do tohoto souboru obvykle nebudete zapisovat příliš mnoho vlastností. Většinou ho použijete jen k definici názvu hostitele pro protokol UUCP. Ten bude implicitně používat název, který nastavíte pomocí příkazu `hostname`, ale je užitečné explicitně nastavit vlastní název pro protokol UUCP. Následuje vzorový soubor `config`:

```
# /usr/lib/uucp/config - hlavní konfigurační soubor UUCP
hostname        vstout
```

Samozřejmě existuje mnoho rozmanitých parametrů, které lze nastavit, příkladem může být název dočasného adresáře nebo přístupová práva k anonymní službě UUCP. Definice přístupových práv k anonymní službě UUCP bude popsána v některé z dalších statí.

12.3.6 Jak sdělit protokolu UUCP informace o jiných systémech – soubor sys

Soubor *sys* popisuje systémy, které zná váš počítač. Záznam je uveden klíčovým slovem *system*; řádky mezi dvěma klíčovými slovy *system* definují parametry, které se vztahují k danému systému. Obecně definuje záznam systému takové parametry, jako je například telefonní číslo a přihlašovací komunikační skript.

Parametry uvedené před prvním výskytem klíčového slova *system* definují implicitní hodnoty, které se budou používat pro všechny systémy. V sekci implicitních hodnot budou obvykle uvedeny parametry protokolu a podobné vlastnosti.

Následuje poměrně podrobný popis nejvýznamnějších polí.

Název systému

Příkaz *system* definuje název vzdáleného systému. Musíte zadat korektní název vzdáleného systému, ne pouze přezdívku, kterou jste si vymysleli, protože program `uucico` porovná tento název s názvem, který pošle vzdálený systém v okamžiku vašeho přihlášení.[7]

Každý název systému se může objevit pouze jedenkrát. Pokud chcete pro daný systém používat několik skupin konfigurací (například různá telefonní čísla, která by měl program `uucico` postupně zkoušet vytáčet), můžete zadat tzv. *zástupce*. Zástupci jsou probíráni dále.

Telefonní číslo

Má-li být vzdálený systém dosažitelný pomocí telefonní linky, určuje pole *phone* číslo, které by měl modem vytočit. Toto pole může obsahovat několik symbolů, které posléze zpracuje vytáčecí procedura programu `uucico`. Symbol rovnítka říká programu `uucico`, aby vyčkal na sekundární vytáčecí tón a symbol pomlčky generuje pauzu dlouhou jednu sekundu. Některé telefonní rozvody se totiž ucpou, když neuděláte pauzu mezi číslem, které musíte vytočit, abyste se dostali z vnitřní sítě, a mezi vlastním telefonním číslem.

Nevím, zda existuje nějaký vhodný pojem, který by toto číslo jednoznačně definoval – ale z vlastní zkušenosti víte, že u některých soukromých telefonních rozvodů uvnitř firmy musíte vytočit číslo 0 nebo 9, a teprve potom se dostanete na vnější telefonní linku.

K utajení informací týkajících se daného systému, jako je telefonní číslo jednotlivých uzlů, můžete použít libovolný řetězec. Každý takovýto řetězec bude pomocí souboru `dialcode` převeden na vytáčecí kód. Předpokládejme, že máte následující soubor `dialcode`:

```
# /usr/lib/uucp/dialcode – převod vytáčecích kódů
Bogoham          024881
Coxton           035119
```

Pomocí těchto definic můžete v souboru *sys* používat například telefonní číslo *Bogoham773*, které snad vypadá přehledněji.

[7] Starší verze 2 protokolu UUCP nevysílá svůj název při volání vzdáleného systému; avšak novější implementace protokolu UUCP tak činí a stejně tak i protokol Taylor UUCP.

Port a rychlost

Volby *port* a *speed* se používají k výběru zařízení, jenž bude používáno k vytáčení vzdáleného systému a k nastavení maximální rychlosti, na kterou může být dané zařízení nastaveno.[8] Záznam *system* může používat buď jednu z těchto voleb, nebo kombinaci obou voleb. Při vyhledávání vhodného zařízení v souboru *port* vybere systém pouze ty porty, které odpovídají názvu portu a/nebo zvolenému rozsahu rychlostí.

Obvykle by měla stačit pouze volbu *speed*. Máte-li v souboru `port` definováno pouze jedno sériové zařízení, zvolí program `uucico` vždy právě toto zařízení, takže pro ně musíte pouze nastavit požadovanou rychlost. Máte-li ke svým systémům připojeno několik modemů, nebudete muset často určovat konkrétní port, protože když program `uucico` zjistí, že danému požadavku vyhovuje více zařízení, bude postupně zkoušet každé z těchto zařízení, dokud nenajde nějaké nepoužívané zařízení.

Přihlašovací komunikační skript

Ve výše uvedeném výkladu jsme se již setkali s přihlašovacím komunikačním skriptem, který sděluje programu `uucico`, jakým způsobem se má přihlásit ke vzdálenému systému. Přihlašovací komunikační skript se skládá ze seznamu symbolů, které definují očekávané řetězce a řetězce posílané místním procesem `uucico`. Cílem je, aby program `uucico` počkal, dokud vzdálený počítač nepošle výzvu k přihlášení. Místní proces pak vrátí přihlašovací jméno, počká až vzdálený systém pošle výzvu k zadání hesla a nakonec pošle vlastní heslo. Očekávané a posílané řetězce se vzájemně střídají. Program `uucico` automaticky přidá ke všem posílaným řetězcům znak návrat vozíku (\r). Jednoduchý komunikační skript by mohl vypadat asi takto:

```
ogin: vstout ssword:  catch22
```

Všimněte si, že pole očekávaných řetězců neobsahují celé výzvy. To proto, aby přihlášení uspělo i v případě, kdy vzdálený systém vyšle místo řetězce `login:` řetězec `Login:`.

Program `uucico` umožňuje i určitý druh podmíněného spouštění, například když je potřeba program `getty` na vzdáleném počítači znovu inicializovat. V tomto případě můžete k očekávanému řetězci připojit komunikační podřetězce, které jsou vzájemně odděleny pomocí pomlček. Komunikační podřetězce se spustí pouze v případě, že selže hlavní očekávaný řetězec, například při překročení časového limitu. Jedním z případů využití této vlastnosti je obdržení kódu BREAK v situaci, kdy vzdálený systém nepošle výzvu k přihlášení. Následující příklad uvádí všeobecný komunikační skript, který bude fungovat i v případě, že před zobrazením výzvy k přihlášení stisknete klávesu Enter. Řetězec „“ sděluje UUCP, aby na nic nečekal a okamžitě pokračoval v posílání řetězce.

8 Přenosová rychlost režimu tty linky musí být nastavena přinejmenším na hodnotu maximální přenosové rychlosti.

```
"" \n\r\d\r\n\c ogin:-BREAK-ogin: vstout ssword: catch22
```

V komunikačním skriptu lze použít spoustu speciálních řetězců a únikových znaků. Následuje neúplný seznam korektních symbolů, které mohou být uvedeny v očekávaných řetězcích:

„“ Prázdný řetězec. Tento řetězec sděluje programu `uucico`, aby na nic nečekal a okamžitě pokračoval v posílání následujícího řetězce.

\t Znak tabulátoru.

\r Znak návrat vozíku (CR).

\s Znak mezera. Tento znak potřebujete k vložení mezer do komunikačního řetězce.

\n Znak nový řádek.

\\ Znak zpětné lomítko.

V posílaných řetězcích se mohou kromě výše uvedených symbolů vyskytnout následující únikové znaky a řetězce:

EOT Znak konce přenosu (`^D`).

BREAK Znak přerušení (Break).

\c Potlačí na konci řetězce poslání znaku návrat vozíku.

\d Přeruší posílání na jednu sekundu.

\E Povolí kontrolu opakování. Tato vlastnost vyžaduje, aby program `uucico` počkal s odezvou tak dlouho, dokud nebudou všechny zapsané informace znovu přečteny z příslušného zařízení. Teprve pak se bude pokračovat ve vykonávání komunikačního skriptu. Tato volba je užitečná zejména při použití modemových komunikačních skriptů (se kterými se setkáme v dalším výkladu). Implicitně je kontrola opakování vypnutá.

\e Vypne kontrolu opakování.

\K Totéž jako symbol *BREAK*.

\p Zastaví chod na zlomek sekundy.

Zástupci

Někdy je vhodné mít k dispozici několik záznamů daného systému, například tehdy, je-li systém dosažitelný pomocí několika modemových linek. U protokolu Taylor UUCP je možné pro jeden systém definovat několik záznamů pomocí tzv. *zástupců*.

Záznam zástupce přebírá veškerá nastavení z hlavního záznamu systému a jsou v něm zadány pouze ty hodnoty, které by měly potlačit implicitní hodnoty záznamu systému nebo které by měly přidat hodnoty k tomuto záznamu. Zástupce je oddělen od hlavního záznamu systému řádkem, který obsahuje klíčové slovo *alternate*.

Chcete-li u hostitele **pablo** používat dvě telefonní čísla, měli byste následujícím způsobem upravit záznam v souboru *sys*:

```
system        pablo
phone         123-456
... další záznamy...
alternate
phone         123-455
```

Když nyní budete volat hostitele **pablo**, pokusí se nejprve program *uucico* vytočit číslo 123-456 a pokud toto volání neuspěje, pokusí se vytočit číslo zástupce. Záznam zástupce přebírá veškerá nastavení z hlavního záznamu systému a potlačuje pouze nastavení telefonního čísla.

Omezení času volání

Protokol Taylor UUCP nabízí množství způsobů pro omezení časů, ve kterých je možné uskutečnit hovory se vzdáleným systémem. Tato omezení můžete zavést buď kvůli omezením, která klade vzdálený hostitel na své služby během pracovní doby, nebo proto, abyste se vyhnuli dobám s vysokými tarifními poplatky. Poznamenejte si, že pomocí volby `-S` nebo `-f` je možné v programu `uucico` vždy potlačit omezení času volání.

Protokol Taylor UUCP implicitně zakazuje spojení v jakémkoliv čase, takže *musíte* v souboru *sys* použít volbu, která povolí nějaký časový rozsah volání. Pokud vás nezajímá omezení času volání, můžete zadat do souboru `sys` volbu *time*, které přiřadíte hodnotu *Any*.

Nejjednodušší způsob omezení času nabízí volba *time*, za níž následuje řetězec tvořený poli den a čas. Pole den může nabývat libovolné kombinace hodnot *Mo*, *Tu*, *We*, *Th*, *Fr*, *Sa*, *Su* (Pondělí až Neděle) nebo hodnot *Any* (kdykoliv), *Never* (nikdy) nebo *Wk* (pracovní dny). Pole čas se skládá ze dvou 24hodinových časových hodnot, které jsou odděleny pomlčkou. Tyto hodnoty udávají časový rozsah, ve kterém se mohou hovory uskutečňovat. Kombinace těchto polí musí být zapsána bez bílého místa. Pomocí čárek může být seskupen libovolný počet denních a časových údajů, například:

```
time            MoWe0300-0730,Fr1805-2000
```

Tento příkaz povoluje volání v pondělí a ve středu v době od 03:00 do 07:30 a v pátek v době od 18:05 do 20:00. Když časové pole překračuje půlnoc, konkrétně *Mo1830-0600*, znamená to, že hovory budou povoleny v pondělí v době od půlnoci do 06:00 a v době od 18:30 do půlnoci.

Speciální řetězce *Any* a *Never* fungují přesně tak, jak naznačují: hovory se mohou uskutečnit kdykoliv, resp. nikdy.

Příkaz `time` může obsahovat volitelný druhý argument, který definuje čas v minutách, po jehož uplynutí se má uskutečnit nový pokus. Když selže pokus o spojení, nepovolí program `uucico` po určitou předem zadanou dobu další pokus o spojení se vzdáleným hostitelem. Program `uucico` implicitně používá exponenciální zpětné schéma, při kterém se s každým opakovaným selháním zvyšuje čas do nového pokusu. Zadáte-li například čas opakování pokusu 5 minut, odmítne program `uucico` zavolat vzdálený systém dříve, než 5 minut po posledním neúspěšném volání.

Příkaz `timegrade` vám umožňuje přidělit plánu maximální priority. Předpokládejme například, že v záznamu *system* máte následující příkazy `timegrade`:

```
timegrade               N Wk1900-0700,SaSu
timegrade               C Any
```

Tyto příkazy umožní přenos úkolů s prioritou C nebo vyšší (pošta se obvykle uchovává s prioritou B nebo C) při každém uskutečněném spojení, zatímco news (které se obvykle uchovávají s prioritou N) budou přenášeny pouze v noci a o víkendech.

Stejně jako u příkazu `time` lze zadat i u příkazu `timegrade` volitelný třetí argument, který definuje čas (v minutách) do nového pokusu.

Zde se však může objevit následující námitka ohledně priorit: Zaprvé se volba *timegrade* vztahuje pouze na soubory, které posílají *vaše* systémy; vzdálený systém může i přesto přenášet cokoliv ho napadne. Chcete-li explicitně definovat požadavek na vzdálený systém, aby posílal pouze úkoly, které mají vyšší než zadanou prioritu, použijte k tomu volbu *call-timegrade*; nemáte však žádnou záruku, že se bude vzdálený systém tímto požadavkem řídit.[9]

Podobně nebude ani pole *timegrade* ověřeno v případě, kdy si vzdálený systém sám zavolá. V takovém případě mu budou poslány všechny připravené úkoly. Vzdálený systém může explicitně požádat váš program `uucico`, aby se omezil pouze na určitou prioritu.

9 Jestliže používá vzdálený systém protokol Taylor UUCP, bude se tímto požadavkem řídit.

12.3.7 Jaká zařízení existují – soubor port

Soubor `port` informuje program `uucico` o dostupných portech. Těmito porty mohou být jak modemové porty, tak i ostatní podporované typy portů, jako jsou přímé sériové linky nebo sockety protokolu TCP.

Stejně jako soubor `sys`, je i soubor `port` složen ze samostatných záznamů, které začínají klíčovým slovem *port*, za kterým následuje název daného portu. Tento název můžete použít v souboru `sys` v příkazu *port*. Název nemusí být jedinečný; existuje-li více portů se shodným názvem, bude program `uucico` zkoušet postupně všechny porty, dokud nenajde port, který se v daném okamžiku nepoužívá.

Za příkazem *port* by měl následovat příkaz *type*, který definuje typ popisovaného portu. Korektní typy jsou *modem*, typ *direct* pro přímá spojení a typ *tcp* pro sockety protokolu TCP. Pokud příkaz *port* chybí, bude implicitní typ portu nastaven na *modem*.

V této stati budeme probírat pouze modemové porty; porty pro protokol TCP a přímé linky budou probrány v dalších statích.

U modemových a přímých portů musíte pomocí příkazu *device* zadat volací zařízení. Obvykle je to název speciálního souboru zařízení, který se nachází v adresáři `/dev`, například `/dev/cua1`.[10]

V případě modemového zařízení určuje záznam portu také typ modemu, který je k danému portu připojen. Různé typy modemů musí být nakonfigurovány odlišným způsobem. Dokonce i modemy, které tvrdí, že jsou kompatibilní se standardem Hayes, nemusí být ve skutečnosti navzájem kompatibilní.

Z tohoto důvodu musíte programu `uucico` sdělit, jak má inicializovat daný modem a jak má vytočit požadované telefonní číslo. Protokol Taylor UUCP uchovává popisy všech vytáčecích zařízení v souboru `dial`. Chcete-li použít některé z těchto zařízení, musíte pomocí příkazu *dialer* zadat název požadovaného vytáčecího zařízení.

Někdy budete chtít používat modem odlišným způsobem podle typu volaného systému. Například některé starší modemy nerozumí tomu, když se nějaký vysokorychlostní modem pokouší spojit rychlostí 14 400 bps; tyto modemy linku prostě zavěsí, místo aby se snažily spojit například rychlostí 9 600 bps. Víte-li, že systém **drop** používá právě takovýto typ hloupého modemu, musíte nastavit svůj modem odlišným způsobem. K tomu potřebujete další záz-

[10] Někteří lidé namísto těchto zařízení používají zařízení `ttyS*`, která jsou určena pouze na volání směrem do vašeho systému.

nam v souboru `port`, který bude definovat odlišný typ vytáčecího zařízení. Nyní můžete přiřadit novému portu odlišný název, například *serial1-slow,* a v souboru `sys` použít v záznamu pro systém **drop** příkaz *port*.

Výhodnější je ale rozdělit porty podle podporovaných rychlostí. Dva záznamy portů mohou pro výše popsanou situaci vypadat následovně:

```
# NakWell modem; spojení na vysoké rychlosti
port          serial1                  # jméno portu
type          modem                    # modem
device        /dev/cua1                # COM2
speed         38400                    # podporovaná rychlost
dialer        nakwell                  # vytáčecí zařízení

# NakWell modem; spojení na nízké rychlosti
port          seriál1                  # jméno portu
type          modem                    # modem
device        /dev/cua1                # COM2
speed         9600                     # podporovaná rychlost
dialer        nakwell-slow             # nezkoušej vysokou rychlost
                                         připojení
```

Záznam systému pro systém **drop** přiřadí portu název `serial1`, ale bude požadovat, aby se používala pouze rychlost 9 600 bps. Potom program `uucico` automaticky vybere druhý záznam portu. Všechny ostatní systémy, které mají v záznamu systému uvedenu rychlost 38 400 bps, budou volány s použitím prvního záznamu portu.

12.3.8 Jak vytočit číslo – soubor dial

Soubor `dial` popisuje způsob, jakým se používají různá vytáčecí zařízení. Protokol UUCP většinou raději komunikuje s vytáčecími zařízeními, než s fyzickými modemy, protože dříve bývalo obvyklé jedno (drahé) automatické vytáčecí zařízení, které obsluhovalo sadu modemů. Dnes má většina modemů zabudovanou podporu pro vytáčení čísel, takže rozdíl je poměrně nevýrazný.

Přesto však mohou odlišná vytáčecí zařízení nebo modemy vyžadovat odlišné konfigurace. Každou z těchto konfigurací popíšete v souboru *dial*. Záznamy v tomto souboru začínají příkazem *dialer*, který přiřazuje název vytáčecímu zařízení.

Kromě této položky je nejdůležitější komunikační skript, který se zadává pomocí příkazu `chat`. Stejně jako tomu bylo u přihlašovacího komunikačního skriptu, skládá se i tento skript ze sekvence řetězců, které program `uucico` posílá na vytáčecí zařízení, a z odpovědí, které

očekává z vytáčecího zařízení. Obecně se tento skript používá k inicializaci modemu do nějakého známého stavu a k vytočení požadovaného čísla. Následující příklad záznamu vytáčecího zařízení ukazuje typický komunikační skript pro modemy kompatibilní s modemovým standardem Hayes:

```
# NakWell modem; spojení na vysoké rychlosti
dialer                nakwell  # jméno vytáčecího zařízení
chat                  "" ATZ OK\r ATH1E0Q0 OK\r ATDT\T CONNECT
chat-fail             BUSY
chat-fail             ERROR
chat-fail             NO\sCARRIER
dtr-toggle            true
```

Komunikační skript modemu začíná řetězcem „“, což je prázdný očekávaný řetězec. Program `uucico` proto okamžitě pošle první příkaz (`ATZ`). Příkaz `ATZ` inicializuje modemy, které jsou kompatibilní se standardem *Hayes*. Pak bude `uucico` čekat, dokud modem nepošle odpověď `OK` a následně další příkaz, který vypne místní opakování znaků. Jakmile modem opět pošle odpověď `OK`, pošle program `uucico` příkaz pro vytáčení (`ATDT`). Úniková sekvence `\T` v řetězci bude nahrazena skutečným telefonním číslem, které systém získá z příslušného záznamu v souboru `sys`. Potom bude program `uucico` čekat, dokud modem nepošle odpověď `CONNECT`, která bude signalizovat, že bylo spojení se vzdáleným systémem úspěšné.

Často se stává, že se modemu nepodaří spojení se vzdáleným systémem, například když už vzdálený systém komunikuje s nějakým jiným systémem a linka je obsazená. V takovém případě vrátí modem nějakou chybovou zprávu, která označuje příčinu neúspěchu. Komunikační skripty modemu nejsou schopny detekovat takovéto zprávy, takže program `uucico` bude čekat na očekávaný řetězec tak dlouho, dokud nedojde k překročení časového limitu. Proto se v log-souboru protokolu UUCP objeví místo pravého důvodu pouze neurčitá zpráva „`timed out in chat script`“ (v komunikačním skriptu došlo k překročení časového limitu).

Avšak protokol Taylor UUCP umožní sdělit tyto hlášky programu `uucico`. Lze to provést pomocí příkazu *chat-fail*, který je uveden ve výše uvedeném výpisu. Když program `uucico` detekuje při spuštění komunikačního skriptu modemu řetězec, který odpovídá neúspěšnému provedení komunikačního skriptu, zruší aktuální volání a zapíše do log-souboru protokolu UUCP chybovou zpráva.

Poslední příkaz v příkladu sděluje protokolu UUCP, aby přepnul linku DTR dříve, než se spustí komunikační skript modemu. Většina modemů může být nakonfigurována tak, aby zavěsili telefonní linku a přešli do příkazového režimu v případě, že zjistí změnu linky DTR.[11]

[11] Některé modemy můžete také nakonfigurovat tak, aby provedly reset, pokud zjistí změnu na lince DTR. Některé z těchto modemů však tuto vlastnost nepodporují a mohou občas zavěsit.

12.3.9 Použití protokolu UUCP v sítích na bázi protokolu TCP

Na první pohled se může zdát použití protokolu UUCP pro přenos dat po sítích na bázi protokolu TCP absurdní, ale v zásadě to není špatný nápad, zvláště při přenosu velkého množství dat typu usenetových news. U spojení na bázi protokolu TCP se zpravidla vyměňují news pomocí protokolu NNTP, s jehož pomocí se o články žádá a jsou odesílány individuálně, bez komprese a bez jakékoliv další optimalizace. Ačkoliv tato technika vyhovuje rozsáhlým systémům s několika souběžnými news, není příliš vhodná pro malé systémy, které získávají news přes pomalá spojení, jako je ISDN. Takové systémy budou obvykle chtít kombinovat výhody protokolu TCP s výhodami, které přináší posílání news ve velkých skupinách, které lze zkomprimovat a pak je přenést s menší režií. Standardní způsob přenášení takových skupin spočívá v použití protokolu UUCP po sítích na bázi protokolu TCP.

V souboru `sys` musíte následujícím způsobem definovat systém, který má být volán pomocí protokolu TCP:

```
system          gmu
address         news.groucho.edu
time            Any
port            tcp-conn
chat            ogin: vstout word: clouseau
```

Příkaz *address* udává IP-adresu hostitele nebo jeho plně kvalifikované doménové jméno. Odpovídající záznam v souboru `port` by měl vypadat asi takto:

```
port            tcp-conn
type            tcp
service         540
```

Záznam uvádí, že spojení pomocí protokolu TCP by mělo být použito tehdy, pokud záznam v souboru `sys` obsahuje hodnotu *tcp-conn,* a že program `uucico` by se měl pokusit spojit se vzdáleným hostitelem pomocí sítě TCP na portu 540. Toto je implicitní číslo portu pro službu UUCP. Místo čísla portu můžete v příkazu `service` uvést symbolický název portu. Číslo portu, které odpovídá tomuto názvu portu, bude vyhledáno v souboru `/etc/services`. Běžný název pro službu UUCP je `uucpd`.

12.3.10 Použití přímého spojení

Předpokládejme, že pro spojení vašeho systému **vstout** s hostitelem **tiny** používáte přímou linku. Stejně jako v případě použití modemu budete muset pro daný systém zapsat do souboru `sys` záznam. Příkaz `port` označuje sériový port, ke kterému je hostitel *tiny* připojen.

```
system              tiny
time                Any
port                direct1
speed               38400
chat                ogin: cathcart word: catch22
```

V souboru `port` musíte popsat sériový port, který se používá pro přímé spojení. Příkaz *dialer* není nutný, protože u přímé linky není třeba vytáčet žádná čísla.

```
port                direct1
type                direct
speed               38400
```

12.4 Co umí a neumí protokol UUCP – nastavování přístupových práv

12.4.1 Spouštění příkazů

Úkolem protokolu UUCP je kopírovat soubory z jednoho systému do druhého a žádat spouštění určitých příkazů na vzdálených hostitelích. Samozřejmě, že jako správce systému budete chtít řídit práva, která budete přidělovat ostatním systémům – rozhodně není dobré povolit na svém systému spouštění libovolných příkazů.

Jedinými příkazy, které na vašem počítači povolí protokol Taylor UUCP spouštět ostatním systémům, jsou implicitně příkazy `rmail` a `rnews`, které se obecně používají pro spojení pomocí protokolu UUCP k výměně elektronické pošty a usenetových news. Implicitní vyhledávací cesta, kterou používá program `uuxqt`, se definuje v době sestavování tohoto programu, ale obvykle obsahuje adresáře `/bin`, `/usr/bin` a *`/usr/local/bin`*. Chcete-li pro konkrétní systém změnit množinu příkazů, použijte k tomu klíčové slovo *commands*, které přidáte do souboru *sys*. Podobně může být pomocí příkazu *command-path* změněna i vyhledávací cesta. Chcete-li například, aby mohl systém **pablo** spouštět společně s příkazy `rmail` a `rnews` i příkaz `rsmtp`, upravte soubor `sys` následovně:[12]

```
system              pablo
...
commands            rmail rnews rsmtp
```

[12] Příkaz `rsmtp` se používá k doručování pošty pomocí hromadného protokolu SMTP. To je popsáno v kapitolách o poště.

12.4.2 Přenosy souborů

Protokol Taylor UUCP také dovoluje velice podrobné nastavení přenosu souborů. Na jedné straně můžete zcela zakázat přenosy souborů z konkrétního systému nebo do konkrétního systému. Stačí jen nastavit volbu *request* na hodnotu *no* a vzdálený systém nebude moci ani přijímat soubory z vašeho systému, ani žádné soubory vašemu systému posílat. Podobně můžete pomocí volby *transfer*, kterou nastavíte na hodnotu *no*, zakázat svým uživatelům přenos souborů do vašeho systému nebo z vašeho systému. Implicitně je jak místním uživatelům, tak i uživatelům ze vzdáleného systému povoleno posílat a přijímat soubory.

Kromě toho můžete nastavit adresáře, do kterých nebo ze kterých mohou být soubory kopírovány. Obvykle budete chtít omezit přístup ze vzdálených systémů pouze jen na jednu hierarchii adresářů a zároveň budete chtít umožnit vašim uživatelům posílání souborů ze svého domovského adresáře. Uživatelé mohou obecně přijímat ze vzdáleného systému pouze soubory z veřejného adresáře protokolu UUCP, což je adresář `/var/spool/uucppublic`. Toto je tradiční místo, kde je možné soubory zpřístupnit veřejnosti; je to velmi podobné serverům FTP na Internetu. Na tento adresář se běžně odkazuje pomocí znaku vlnka.

Z toho důvodu poskytuje protokol Taylor UUCP čtyři odlišné příkazy, které se používají ke konfiguraci adresářů pro přijímání a posílání souborů. Jsou to konkrétně příkazy *local-send*, který specifikuje seznam adresářů, z nichž může uživatel stahovat soubory pomocí protokolu UUCP, dále příkaz *local-receive*, který uvádí seznam adresářů, do nichž může uživatel přijímat soubory pomocí protokolu UUCP, a příkazy *remote-send* a *remote-receive*, které se chovají analogicky při požadavku z cizího systému. Uvažte následující příklad:

```
system              pablo
...
local-send          /home
local-receive       /home ~/receive
remote-send         ~ !~/incoming !~/receive
remote-receive      ~/incoming
```

Příkaz *local-send* povoluje uživatelům posílat z vašeho hostitele na hostitele **pablo** libovolné soubory z adresářové struktury pod adresářem `home` a z veřejného adresáře protokolu UUCP. Příkaz *local-receive* povoluje těmto uživatelům přijímat soubory buď do volně dostupného adresáře `receive`, do něhož lze zapisovat a který se nachází pod adresářem `uucppublic`, nebo do libovolného adresáře s možností zápisu, který se nachází v adresářové struktuře pod adresářem `/home`. Příkaz *remote-send* povoluje uživatelům z hostitele **pablo** žádat o soubory z adresáře `/var/spool/uucppublic`, vyjma souborů z adresářových struktur pod ad-

resáři `incoming` nebo `receive`. Tyto vyjmuté adresáře jsou signalizovány programu `uucico` tak, že před jejich názvy je uveden vykřičník. A konečně poslední řádek povoluje uživatelům z hostitele **pablo** přenášet libovolné soubory do adresáře `incoming`.

Jedním z největších problémů souvisejících s přenosem souborů prostřednictvím protokolu UUCP je skutečnost, že lze přijímat soubory pouze do adresářů, do kterých lze zapisovat a které jsou volně dostupné. To může svádět některé uživatele k tomu, aby chystali pastičky na ostatní uživatele. Tomuto ale nelze nijak zabránit, ledaže byste zablokovali veškeré přenosy souborů pomocí protokolu UUCP.

12.4.3 Doručování

Protokol UUCP nabízí mechanismus, pomocí něhož můžete na ostatních systémem spustit přenosy souborů pod svým jménem. Například následující mechanismus umožňuje, aby pro vás hostitel **seci** přijal soubor z hostitele **uchile** a poslal ho do vašeho systému.

```
$ uucp -r seci!uchile!~/find-ls.gz ~/uchile.filez.gz
```

Tato technika, pomocí níž lze předávat úkol přes několik systémů, se nazývá *doručování*. Ve výše uvedeném příkladu může být důvodem použití doručovací techniky například to, že hostitel **seci** má přístup pomocí protokolu UUCP k hostiteli **uchile**, zatímco váš hostitel nemá přístup k tomuto hostiteli. Avšak pokud provozujete systém s protokolem UUCP, budete chtít omezit doručovací službu pouze na několik důvěryhodných hostitelů, kteří vám nevyženou telefonní účet do neúnosných mezí například tím, že by chtěli po vašem hostiteli stáhnout poslední verzi zdrojového kódu balíku X11R6.

Implicitně má protokol Taylor UUCP zablokované veškeré doručování. Chcete-li povolit doručování nějakému konkrétnímu systému, použijte k tomu příkaz *forward*. Tento příkaz udává seznam systémů, ze kterých a na které je možné doručovat úkoly. O doručování úkolů vás může požádat nějaký další systém. Například správce protokolu UUCP na hostiteli **seci** bude chtít přidat do souboru *sys* následující řádky, které povolí hostiteli **pablo** požadovat soubory po hostiteli **uchile**:

```
####################
# pablo
system                pablo
...
forward               uchile
####################
# uchile
```

```
system                  uchile
...
forward-to              pablo
```

Položka *forward-to* je pro hostitele **uchile** nutná, aby jakékoliv soubory poslané z tohoto hostitele byly předány hostiteli **pablo**. V opačném případě by je protokol UUCP zahodil. Tato položka je obměnou příkazu *forward* a hostiteli **uchile** povoluje posílat soubory pouze na hostitele **pablo** přes hostitele **seci**; jakákoliv jiná cesta je nepřípustná.

Chcete-li povolit doručování do libovolného systému, použijte k tomu speciální klíčové slovo *ANY* (vyžadují se velká písmena).

12.5 Nastavení systému pro příchozí volání

Chcete-li nastavit svůj systém pro příchozí volání, musíte u svých sériových portů povolit přihlášení a přizpůsobit některé ze systémových souborů tak, aby poskytovaly účty pro protokol UUCP. To vše bude náplní této stati.

12.5.1 Nastavení programu getty

Pokud chcete používat sériovou linku jako port pro příchozí volání, musíte na tomto portu povolit proces `getty`. Bohužel některé implementace programu `getty` nejsou pro tento účel vhodné, protože budete většinou chtít používat sériový port jak pro volání ze systému, tak pro volání směrem do systému. Proto se musíte ujistit, že používáte takový program `getty`, který je schopen sdílet linku s dalšími programy, jako jsou například programy `uucico` nebo `minicom`. Jeden z programů, který tyto požadavky splňuje, je program `uugetty` z balíku `getty_ps`. Většina linuxových distribucí tento program obsahuje; zkontrolujte, zda máte program `uugetty` ve svém adresáři `/sbin`. Pak je tu ještě jeden program od Gerta Doeringa s názvem `mgetty`, který navíc podporuje příjem faxů. Nejnovější verze těchto programů můžete získat buď v podobě binárních programů, nebo jako zdrojové kódy na adrese **sunsite.unc.edu**.

Vysvětlení odlišností ve způsobu, jakým se tyto programy starají o přihlášení, přesahuje rámec této malé stati; chcete-li více informací, nahlédněte prosím do dokumentu Serial HOWTO od Gregha Hankinse a také do dokumentace, která přichází společně s balíky `getty_ps` a `mgetty`.

12.5.2 Poskytování účtů protokolu UUCP

Dále budete muset nastavit uživatelské účty, které umožní vzdáleným systémům přihlášení do vašeho systému a uskutečnění spojení pomocí protokolu UUCP. Zpravidla poskytnete každému systému, s nímž jste v kontaktu, samostatné přihlašovací jméno. Když budete nastavovat účet pro hostitele **pablo**, přiřadíte mu asi jméno uživatele **Upablo**.

Účty systémů, které k vám volají po sériovém portu, musíte obvykle přidat do systémového souboru hesel `/etc/passwd`. Osvědčilo se vložit všechna přihlášení pomocí protokolu UUCP do speciální skupiny, například s názvem **uuguest**. Domovský adresář daného účtu by měl být nastaven do veřejného dočasného adresáře `/var/spool/uucppublic`; jeho přihlašovací rozhraní musí tvořit program `uucico`.

Jestliže máte nainstalovaný balík se stínovými hesly, můžete k tomuto nastavení použít příkaz `useradd`:

```
# useradd -d /var/spool/uucppublic -G uuguest
  -s /usr/lib/uucp/uucico Upablo
```

Jestliže nepoužíváte balík se stínovými hesly, budete pravděpodobně muset ručně upravit soubor `/etc/passwd`, přidat do něj níže uvedenou řádku, kde hodnoty 5 000 a 150 jsou číselná uid a gid, která jsou přidělena uživateli **Upablo**, resp. skupině **uuguest**.

```
Upablo:x:5000:150:UUCP Account:/var/spool/uucppublic:
  /usr/lib/uucp/uucico
```

Po nainstalování účtu musíte tento účet zaktivovat, to lze provést nastavením hesla pomocí příkazu *passwd*.

Abyste mohli obsluhovat systémy UUCP, které se připojují k vašemu systému pomocí sítě na bázi protokolu TCP, musíte nastavit super-server `inetd`, jenž se bude starat o přicházející spojení na portu *uucp*. To lze provést přidáním následující řádky do souboru `/etc/inetd.conf`:[13]

```
uucp    stream  tcp    nowait  root
  /usr/sbin/tcpd  /usr/lib/uucp/uucico -l
```

Parametr `-l` sděluje programu `uucico`, aby prováděl při přihlášení vlastní proceduru ověření totožnosti. Program `uucico` vyzve, stejně jako standardní program `login`, k zadání přihlašovacího jména a hesla, avšak bude se spoléhat pouze na svoji vlastní soukromou databázi, a nikoliv na soubor `/etc/passwd`. Tento soukromý soubor s hesly se nazývá `/usr/lib/uucp/passwd` a obsahuje páry tvořící přihlašovací jména a hesla:

13 Pamatujte, že démon `tcpd` běží obvykle v režimu 700, takže ho musíte vyvolat jako uživatel **root** a nikoliv jako uživatel **uucp**, což byste za normálních okolností asi udělali.

```
Upablo  IslaNegra
Ulorca  co'rdoba
```

Samozřejmě tento soubor musí vlastnit uživatel **uucp** a musí mu být přidělena přístupová práva 600.

Jestliže vám připadá tato databáze jako velmi dobrá myšlenka, kterou byste chtěli uplatnit i u standardních sériových přihlašovacích procedur, budete asi zklamáni, když vám řeknu, že to momentálně nepřipadá v úvahu, aniž byste provedli zásadní úpravy. Zaprvé k tomu budete potřebovat protokol Taylor UUCP verze 1.05, protože ten umožňuje, aby program `getty` pomocí volby `-u` podstoupil jméno volajícího uživatele programu `uucico`.[14] Potom musíte obelstít vámi používaný program `getty`, aby namísto běžného vyvolání programu `/bin/login` vyvolal program `uucico`. U balíku `getty_ps` to lze provést přidáním volby *LOGIN* do konfiguračního souboru. Avšak tato volba úplně zakáže interaktivní přihlašování. Na druhou stranu má balík `mgetty` krásnou vlastnost, která vám umožní na základě poskytnutého jména uživatele vyvolat odlišné přihlašovací příkazy. Například můžete programu `mgetty` sdělit, aby používal program `uucico` u všech uživatelů, jejichž přihlašovací jméno začíná písmenem velké U a u všech ostatních jmen uživatelů bude program `mgetty` používat standardní příkaz `login`.

Abyste chránili své uživatele protokolu UUCP od volajících, kteří udávají falešný název systému a kteří sledují veškerou poštu vašich uživatelů, měli byste ke každému záznamu systému v souboru `sys` přidat příkaz *called-login*. Ten je popsán v následující stati Jak se chráníme před podvodníky.

12.5.3 Jak se chráníme před podvodníky

Jeden z největších problémů s protokolem UUCP spočívá v tom, že volající systém může lhát o svém názvu; on sice po přihlášení ohlásí svůj název volanému systému, ale server nemá žádnou možnost, jak by ho mohl ověřit. Takhle se může útočník přihlásit k jeho nebo k jejímu účtu protokolu UUCP a předstírat, že je někdo jiný a může si z tohoto systému vyzvednout poštu. To je zejména problematické v situaci, kdy nabízíte přihlášení pomocí anonymní služby UUCP, jejíž heslo je veřejně dostupné.

Dokud si nejste jisti, že můžete věřit všem systémům volajícím na váš systém, že jsou poctivé, *musíte* se jistit před tímto druhem podvodníků. Lék na tuto nemoc spočívá v tom, že budete po každém systému vyžadovat, aby používal konkrétní přihlašovací jméno, což lze provést uvedením volby *called-login* v souboru `sys`. Vzorový záznam pro daný systém by mohl vypadat asi takto:

[14] Parametr `-u` je také dostupný ve verzi 1.04, avšak nemá přiřazenu žádnou funkci.

```
system           pablo
... obvyklé volby ...
called-login     Upablo
```

Výsledkem je, že kdykoliv se systém přihlásí a bude předstírat, že je hostitel **pablo**, zkontroluje program `uucico`, zda se tento systém přihlásil jako uživatel **Upablo**. Jestliže se volající systém nepřihlásil jako tento uživatel, bude odmítnut a spojení bude zavěšeno. Měli byste si zvyknout přidat volbu *called-login* ke každému záznamu systému, který přidáte do souboru `sys`. Je důležité, abyste toto opatření provedli u *všech* systémů, bez ohledu na to, zda se již s vaším systémem spojily, či nikoliv. U těch systémů, jež vás ještě nikdy nevolaly, byste měli nastavit volbu *called-login* na nějaké úplně umělé jméno uživatele, například na jméno **neverlogsin**.

12.5.4 Buďte paranoidní – sekvenční kontrola hovorů

Další způsob, který může detekovat a odrazit podvodníky, představuje použití sekvenčních kontrol hovorů. Sekvenční kontrola hovorů vám může pomoci s ochranou před vetřelci, kteří si nějakým způsobem opatřili přihlašovací heslo vašeho systému s protokolem UUCP.

Při použití sekvenční kontroly hovorů si oba počítače uchovávají záznamy o počtu v minulosti uskutečněných hovorů. S každým spojením se tento počet zvyšuje. Po procesu přihlášení vyšle volající svůj sekvenční počet hovorů a volaný si tento počet zkontroluje se svým vlastním počtem uskutečněných hovorů. Jestliže si počty neodpovídají, bude pokus o uskutečnění spojení odmítnut. Jestliže je základní počet zvolen jako náhodné číslo, stráví útočníci dlouhou dobu, než se jim povede uhodnout správný sekvenční počet hovorů.

Avšak sekvenční kontrola hovorů toho dokáže ještě více: Dokonce i když se nějaké velmi chytré osobě povede detekovat váš sekvenční počet hovorů společně s vaším heslem, stejně na to přijdete. Když nějaký útočník zavolá na systém UUCP vašeho zásobitele a ukradne vám poštu, zvýší se u vašeho zásobitele sekvenční číslo hovoru o jedničku. Když se v budoucnu budete chtít spojit s vaším zásobitelem a pokusíte-li se přihlásit, vzdálený program `uucico` vás odmítne, protože čísla se v žádném případě nebudou shodovat!

Jestliže máte povolenou sekvenční kontrolu hovorů, měli byste pravidelně sledovat soubory log, zdali se v nich nevyskytují chybové hlášky, jež by mohly naznačovat možné útoky. Jestliže váš systém odmítne sekvenční počet hovorů, který poskytne volající systém, vloží program `uucico` hlášku do log-souboru, která říká něco ve smyslu „`Out of sequence call rejected`“ (odmítnuto z důvodu chybného sekvenčního počtu hovorů). Jestliže je váš systém odmítnut svým zásobitelem, protože sekvenční čísla nejsou synchronní, vloží do souboru log hlášku „`Handshake failed (RBADSEQ)`“ (neuspěla inicializační fáze (RBADSEQ)).

Chcete-li povolit sekvenční kontrolu, musíte přidat do záznamu systému následující řádku:

```
# povolení sekvenční kontroly hovorů
sequence         true
```

Kromě toho musíte vytvořit soubor obsahující vlastní sekvenční čísla. Protokol Taylor UUCP uchovává sekvenční čísla v souboru s názvem `.Sequence`, jenž se nachází v dočasném adresáři vzdáleného hostitele. Vlastníkem tohoto souboru *musí* být uživatel **uucp** a musí mu být přiřazen režim 600 (to znamená, že do něj může zapisovat a z něho může číst pouze uživatel **uucp**). Nejlepší je tento soubor založit s libovolnou, předem dohodnutou počáteční hodnotou. V opačném případě se může útočníkovi povést uhodnout skutečný počet hovorů například tak, že vyzkouší všechny hodnoty menší než číslo 60.

```
# cd /var/spool/uucp/pablo
# echo 94316 > .Sequence
# chmod 600 .Sequence
# chown uucp.uucp .Sequence
```

Samozřejmě, že vzdálený systém musí zároveň povolit sekvenční kontrolu hovorů a ta musí mít nastaven sekvenční počet hovorů na přesně stejnou hodnotu, na jakou ho máte nastaven vy.

12.5.5 Anonymní přístup pomocí protokolu UUCP

Chcete-li poskytovat anonymní přístup pomocí protokolu UUCP, musíte nejprve dříve opsaným způsobem nastavit speciální účet pro tuto službu. Běžně se tomuto účtu přiřazuje stejné přihlašovací jméno a heslo, například **uucp**.

Kromě toho musíte ještě pro neznámé systémy nastavit několik bezpečnostních voleb. Můžete jim například zakázat spouštění jakýchkoliv příkazů na vašem systému. Avšak tyto parametry nemůžete nastavit v záznamu v souboru `sys`, protože příkaz *system* vyžaduje název systému, který vy zatím neznáte. Protokol Taylor UUCP řeší tento problém pomocí dalšího příkazu *unknown*. Příkaz *unknown* lze použít v souboru `config` ke specifikaci libovolného příkazu, který by za normálních okolností byl uveden v záznamu systému:

```
unknown          remote-receive ~/incoming
unknown          remote-send ~/pub
unknown          max-remote-debug none
unknown          command-path /usr/lib/uucp/anon-bin
unknown          commands rmail
```

První dva řádky povolí neznámým systémům stahování souborů pouze z adresářové struktury, která se nachází pod adresářem `pub`, a posílání souborů povolí pouze do adresáře `incoming`, který se nachází pod adresářem `/var/spool/uucppublic`. Další řádek sdělí pro-

gramu `uucico`, aby ignoroval veškeré požadavky ze vzdáleného systému, které by požadovaly zapnutí místního ladění. Poslední dva řádky povolují neznámým systémům spouštět příkaz `rmail`; avšak cesta uvedená u tohoto příkazu sděluje programu `uucico`, aby hledal příkaz `rmail` pouze v soukromém adresáři s názvem `anon-bin`. To vám umožní poskytovat speciální příkaz `rmail`, který bude například umět doručit superuživateli veškerou poštu k prozkoumání. Tento příkaz současně umožní anonymním uživatelům zastihnout správce systému, ale zabrání jim vložit jakoukoliv poštu, která je určena pro jiné systémy.

Abyste povolili anonymní službu UUCP, musíte zadat do souboru `config` minimálně jeden příkaz *unknown*. V opačném případě by program `uucico` odmítl všechny anonymní systémy.

12.6 Nízkoúrovňové protokoly protokolu UUCP

Ke sjednání řízení relace a přenosu souborů se vzdáleným hostitelem používá program `uucico` skupinu standardizovaných zpráv. Tato skupina zpráv se často označuje jako tzv. vysokoúrovňový protokol. Během inicializační fáze a během fáze zavěšování se tyto zprávy posílají jako řetězce. Avšak během přenosové fáze se používá doplňkový nízkoúrovňový protokol, který je většinou pro vyšší úrovně transparentní. To proto, aby se například při použití nespolehlivých linek mohla provádět detekce chyb.

12.6.1 Celkový pohled na protokol

Protože se protokol UUCP používá pro rozdílné typy spojení, jako jsou sériové linky nebo sítě na bázi protokolu TCP nebo dokonce X.25, jsou zapotřebí nízkoúrovňové protokoly. Kromě toho různé implementace protokolu UUCP obsahují odlišné protokoly, které v zásadě provádějí stejné služby.

Protokoly se mohou rozčlenit do dvou kategorií: na protokoly interaktivní a na protokoly používající pakety. Protokoly interaktivní přenášejí soubor jako celek, eventuálně mohou u tohoto souboru vypočítat kontrolní součet. Tento typ nemá žádnou režii, avšak vyžaduje spolehlivé spojení, protože jakákoliv chyba způsobí, že soubor musí být celý poslán znovu. Tyto protokoly se zpravidla používají u spojení na bázi protokolu TCP, avšak jejich použití není vhodné u telefonních linek. I když moderní modemy odvádějí poměrně dobrou práci, co se týká korekce chyb, přesto nejsou perfektní a navíc ani neexistuje žádná detekce chyb mezi vaším počítačem a modemem.

Na druhou stranu protokoly používající pakety rozdělují soubor na několik stejně velikých kousků. Každý paket je posílán a přijímán odděleně, je vypočítáván kontrolní součet a zasílateli je podáváno potvrzení o přijetí. Aby se doprava ještě více zefektivnila, byly vynalezeny protokoly s proměnlivými okny, které připouštějí v libovolném okamžiku omezený počet (ok-

no) nevyřízených potvrzení. To značně sníží množství času, které stráví program `uucico` během přenosu vyčkáváním. Navíc jsou tyto protokoly neefektivní u sítí na bázi protokolu TCP, protože mají poměrně velkou režii v porovnání s interaktivními protokoly.

Rozdíl je i v šířce přenosové cesty. Někdy je po sériovém spojení nemožné poslat osmibitové znaky, například když spojení prochází přes hloupý terminálový server. V takovém případě musí být při posílání znaky z osmibitové sady uvozeny. Když posíláte osmibitové znaky po sedmibitovém spojení, musí tyto znaky projít konverzí, která zdvojnásobí množství přenášených dat, i když toto množství dat může být nakonec určitým způsobem kompenzováno interní hardwarovou kompresí. Linky, jež jsou schopny přenášet libovolné osmibitové znaky, se zpravidla nazývají přímé osmibitové linky. To je případ všech spojení na bázi protokolu TCP, stejně tak je tomu i u většiny modemových spojení.

V protokolu Taylor UUCP verze 1.04 jsou dostupné následující nízkoúrovňové protokoly:

g Toto je nejběžnější protokol a měly by mu rozumět snad všechny programy `uucico`. Protokol *g* umí důkladnou detekci chyb, a proto je vhodný zejména pro nekvalitní telefonní linky. Protokol *g* vyžaduje přímé osmibitové spojení. Je to paketově orientovaný protokol používající techniku proměnlivých oken.

i Toto je obousměrný protokol používající pakety, protokol, který může současně posílat a přijímat soubory. Tento protokol vyžaduje spojení s obousměrným přenosem dat a přímou osmibitovou datovou cestu. Tento protokol zvládá momentálně pouze protokol Taylor UUCP.

t Tento protokol je navržen pro použití u spojení na bázi protokolu TCP nebo u jiných skutečně bezchybných sítí. Používá pakety o velikosti 1 024 bajtů a vyžaduje přímé osmibitové spojení.

e Tento protokol dělá v zásadě totéž, co protokol *t*. Hlavní odlišnost spočívá v tom, že protokol *e* je interaktivní protokol.

f Tento protokol byl navržen pro použití u spolehlivých spojení X.25. Je to interaktivní protokol a očekává sedmibitovou datovou cestu. Osmibitové znaky jsou uvozeny, což může být značně neefektivní.

G Toto je verze protokolu *g* z balíku System V verze 4. Tento protokol také zvládají některé další verze protokolu UUCP.

a Tento protokol je podobný protokolu *ZMODEM*. Vyžaduje osmibitové spojení, avšak uvozuje určité kontrolní znaky, jako je XON a XOFF.

12.6.2 Nastavení přenosového protokolu

Všechny protokoly umožňují určitou změnu velikosti paketů, změnu hodnot překročení časového limitu atp. Ve standardních podmínkách obvykle fungují implicitní hodnoty dobře, avšak ve vaší situaci nemusí být optimální. Například protokol *g* může používat velikosti oken v rozmezí od 1 do 7 a velikosti paketů, jako násobky čísla 2, v rozsahu od 64 do 4 096.[15] Jestliže je vaše telefonní linka většinou rušená tak, že se po ní ztrácí více než 5 procent všech paketů, měli byste asi snížit velikost paketu a zmenšit velikost okna. Na druhou stranu, u velmi kvalitních telefonních linek vám může připadat režie protokolu příliš nehospodárná, protože protokol bude posílat potvrzení ACK u každých 128 bajtů. V tomto případě můžete chtít zvýšit velikost paketu na hodnotu 512 bajtů nebo dokonce na 1 024 bajtů.

Protokol Taylor UUCP poskytuje mechanismus, který může vyhovět vašim potřebám. Všechny tyto parametry lze nastavit pomocí příkazu *protocol-parameter*, jenž se umísťuje do souboru *sys*. Například chcete-li pro komunikaci s hostitelem **pablo** nastavit u protokolu *g* velikost paketu na 512 bajtů, přidejte do souboru *sys* následující řádku:

```
system           pablo
...
protocol-parameter g  packet-size  512
```

Nastavitelné parametry a jejich názvy se liší protokol od protokolu. Chcete-li kompletní seznam těchto parametrů, nahlédněte prosím do dokumentace, která je součástí zdrojového kódu protokolu Taylor UUCP.

12.6.3 Výběr konkrétních protokolů

Ne každá implementace programu `uucico` komunikuje a rozumí každému protokolu, takže během inicializační fáze se musí oba procesy dohodnout na použití společného protokolu. Řídicí program `uucico` nabídne řízenému programu `uucico` seznam podporovaných protokolů (pošle řetězec `Pprotlist`), ze kterého si může řízený program vybrat nějaký protokol.

Na základě typu používaného portu (modem, protokol TCP nebo přímá linka) sestaví program `uucico` implicitní seznam protokolů. U modemových a přímých spojení bude tento seznam zpravidla obsahovat protokoly *i*, *a*, *g*, *G* a *j*. U spojení na bázi protokolu TCP bude seznam obsahovat protokoly *t*, *e*, *i*, *a*, *g*, *G*, *j* a *f*. Tento seznam můžete potlačit pomocí příkazu *protocols*, jenž může být uveden buď v záznamu systému, nebo v záznamu portu. Například v souboru *port* můžete upravit záznam vašeho modemového portu následujícím způsobem:

15 Většina binárních souborů, které jsou součástí standardních distribucí Linuxu, mají nastavenou implicitní velikost okna na hodnotu 7 a velikost paketů na 128 bajtů.

```
port              serial1
...
protocols         igG
```

Tento příkaz bude vyžadovat, aby veškerá přicházející a odcházející spojení na tomto portu používala protokoly *i*, *g* nebo *G*. Jestliže vzdálený systém nepodporuje ani jeden z těchto protokolů, pak bude rozhovor neúspěšný.

12.7 Hledání a odstraňování problémů

Tato stať popisuje to, co by mohlo zlobit u vašeho spojení pomocí protokolu UUCP a dává vám doporučení, kde byste měli hledat chyby. Avšak otázky byly sestaveny přímo z hlavy. Existuje mnohem více věcí, které by mohly dělat problémy.

V každém případě povolte ladění pomocí volby `-xall` a podívejte se na výpis v adresáři `Debug`, který se nachází ve vašem dočasném adresáři. Tento výpis by vám měl rychle pomoci rozpoznat, v čem problém vězí. Taktéž se mi osvědčilo zapnout reproduktor u mého modemu v případě, že jsem se nemohl spojit se vzdáleným systémem. U modemů kompatibilních se standardem Hayes to lze provést přidáním řetězce „`ATL1M1 OK`“ do komunikačního skriptu modemu, jenž se nachází v souboru `dial`.

Nejprve byste měli vždy zkontrolovat, zda jsou správně nastavena všechna práva u souborů. Program `uucico` by měl mít nastaveno uid na **uucp** a všechny soubory v adresářích `/usr/lib/uucp`, `/var/spool/uucp` a `/var/spool/uucppublic` by měly být vlastněny uživatelem **uucp**. V dočasném adresáři existuje také několik skrytých souborů[16], které musí vlastnit uživatel **uucp**.

Program `uucico` vypíše „`Wrong time to call`“: Tato hláška obvykle znamená, že jste buď v záznamu systému v souboru `sys` nezadali příkaz *time*, který určuje, v jakých časech se můžete spojit se vzdáleným systémem, nebo jste v něm zadali takový příkaz `time`, jenž v současné době zakazuje volání. Jestliže není poskytnut žádný časový plán volání, bude program `uucico` předpokládat, že se nedá se systémem nikdy spojit.

Program `uucico` si stěžuje, že daný systém je již zablokován: To znamená, že program `uucico` detekoval pro daný vzdálený systém zamykací soubor v adresáři `/var/spool/uucp`. Zamykací soubor může zůstat z dřívějšího spojení se systémem, které zhavarovalo nebo které bylo zrušeno. Ale je také pravděpodobné, že by mohl existovat ještě jeden proces `uucico`, který se snažil dovolat na vzdálený systém a zůstal viset někde v komunikač-

15 To znamená soubory, jejichž názvy začínají tečkou. Takové soubory nejsou příkazem *ls* standardně zobrazeny.

ním skriptu apod. Jestliže se procesu `uucico` nepodaří úspěšně spojit se vzdáleným systémem, ukončete tento proces vysláním zavěšujícího signálu a odstraňte všechny zamykací soubory, které po tomto procesu zůstaly.

Mohu se spojit se vzdáleným systémem, ale komunikační skript nefunguje správně: Podívejte se na text, jenž obdržíte ze vzdáleného systému. Jestliže je zkomolený, může to naznačovat, že jde o problém s rychlostí. V opačném případě si ověřte, že obdržený text skutečně souhlasí s textem, který očekává váš komunikační skript. Pamatujte si, že komunikační skript vždy začíná očekávaným řetězcem. Jestliže obdržíte výzvu k přihlášení a pošlete svoje uživatelské jméno, ale už nikdy neobdržíte výzvu k zadání hesla, vložte mezi tyto řetězce nějakou prodlevu, nebo dokonce můžete vložit prodlevu mezi jednotlivé znaky. V tomto případě byste mohli být rychlejší, než je váš modem.

Můj modem nevytáčí: Jestliže váš modem neindikuje vzestup linky DTR, když se váš program `uucico` snaží volat směrem ze systému, pravděpodobně jste programu `uucico` nesdělili správné zařízení. Jestliže váš modem rozeznává linku DTR, ověřte si pomocí terminálového programu, že na toto zařízení můžete posílat znaky. Jestliže posílání znaků funguje, zapněte na začátku komunikačního skriptu modemu opakování znaků pomocí direktivy *E*. Jestliže se ani po zapnutí opakování znaků nebudou během komunikačního skriptu modemu zobrazovat vaše příkazy, zkontrolujte, zda není na váš modem rychlost linky příliš vysoká nebo nízká. Jestliže vidíte opakování znaků, zkontrolujte, zda jste nevypnuli odpovědi modemu nebo zda jste je nenastavili na číselné kódy. Zkontrolujte, zda je vlastní komunikační skript správný. Pamatujte si, že pokud chcete poslat na modem zpětné lomítko, musíte za sebou napsat dvě zpětná lomítka.

Můj modem se pokouší volat, ale nemůže se dostat na vnější telefonní síť: Do telefonního čísla vložte prodlevu. To je zejména užitečné v případě, kdy se snažíte volat z vnitřní telefonní sítě společnosti. Lidé z Evropy, kteří vytáčejí čísla pomocí pulsní volby, by měli zkusit tónovou volbu. V některých zemích teprve nedávno zlepšili telekomunikační společnosti své sítě. V těchto případech by mohla pomoci tónová volba.

V log-souboru je uvedeno, že mám extrémně vysokou ztrátu paketů: To vypadá na problém s rychlostí komunikace. Že by spojení mezi počítačem a modemem bylo příliš pomalé (pamatujte si, že byste toto spojení měli nastavit na nejvyšší možnou efektivní rychlost)? Nebo, že by váš hardware byl tak pomalý a nedokázal by obsloužit včas přerušení? Říká se, že sériový port s čipovou sadou NSC 16550A pracuje docela dobře při rychlosti 38 400 bps; avšak bez vyrovnávací paměti FIFO (například u čipů 16 450) je nejvyšší možná rychlost 9 600 bps. Také byste se měli ujistit, že máte u vaší sériové linky povoleno hardwarové řízení toku dat.

Další pravděpodobnou příčinou by mohlo být to, že nemáte na portu povoleno hardwarové řízení toku dat. Protokol Taylor UUCP verze 1.04 neobsahuje žádné prostředky na nastavení řízení toku dat na RTS/CTS. Toto řízení musíte explicitně povolit v souboru *rc.serial* pomocí následující řádky:

```
$ stty crtscts < /dev/cua3
```

Mohu se přihlásit, avšak inicializační fáze neuspěje: Zde se může vyskytovat spousta problémů. Hodně by vám měly pomoci záznamy uvedené v souboru `log`. Podívejte se na seznam protokolů, které nabízí vzdálený systém (vzdálený systém pošle během inicializační fáze řetězec `Pprotlist`). Možná ani jeden z protokolů nemají oba systémy společný (zvolili jste vůbec nějaký protokol v souboru `sys` nebo `port`?).

Jestliže vzdálený systém pošle řetězec `RLCK`, v tom případě na vzdáleném systému existuje pro váš systém starý zamykací soubor. Jestliže to není z důvodu, že jste již spojeni se vzdáleným systémem na nějaké jiné lince, požádejte o jeho odstranění.

Jestliže vzdálený systém pošle řetězec `RBADSEQ`, v tom případě je na vzdáleném systému zapnuta sekvenční kontrola hovorů, avšak sekvenční počty hovorů si neodpovídají. Jestliže vzdálený systém pošle řetězec `RLOGIN`, nebylo vám povoleno přihlášení pod tímto číslem id.

12.8 Log-soubory

Pokud byl váš balík UUCP sestaven se zapisováním do log-souborů kompatibilních s protokolem Taylor UUCP, budete mít pouze tři globální log-soubory, které budou umístěny v dočasném adresáři. Hlavní log-soubor se nazývá `Log` a obsahuje všechny informace o uskutečněných spojení a o přenesených souborech. Typický výpis z tohoto souboru vypadá asi následovně (po malém přeformátování, aby se vešel na šířku stránky):

```
uucico pablo - (1994-05-28 17:15:01.66 539)
  Calling system pablo (port cua3)
uucico pablo - (1994-05-28 17:15:39.25 539) Login successful
uucico pablo - (1994-05-28 17:15:39.90 539) Handshake successfull
                (protocol 'g' packet size 1024 window 7)
uucico pablo postmaster (1994-05-28 17:15:43.65 539)
  Receiving D.pabloB04aj
uucico pablo postmaster (1994-05-28 17:15:46.51 539)
  Receiving X.pabloX04ai
uucico pablo postmaster (1994-05-28 17:15:48.91 539)
  Receiving D.pabloB04at
```

```
uucico pablo postmaster (1994-05-28 17:15:51.52 539)
  Receiving X.pabloX04as
uucico pablo postmaster (1994-05-28 17:15:54.01 539)
  Receiving D.pabloB04c2
uucico pablo postmaster (1994-05-28 17:15:57.17 539)
  Receiving X.pabloX04c1
uucico pablo - (1994-05-28 17:15:59.05 539)
  Protocol 'g' packets: sent 15,
                 resent 0, received 32
uucico pablo - (1994-05-28 17:15:16.02 539) Call complete
uuxq pablo postmaster (1994-05-28 17:16:11.11 546)
  Executing X.pabloX04ai
                (rmail okir)
uuxq pablo postmaster (1994-05-28 17:16:13.30 546)
  Executing X.pabloX04as
                (rmail okir)
uuxq pablo postmaster (1994-05-28 17:16:13.51 546)
  Executing X.pabloX04c1
                (rmail okir)
```

Další důležitý log-soubor se nazývá `Stats` a jsou v něm vypsány statistiky přenesených souborů. Část ze souboru `Stats`, která odpovídá výše uvedenému přenosu, vypadá asi takto:

```
postmaster pablo (1994-05-28 17:15:44.78)
              received 1714 bytes in 1.802 seconds (851 bytes/sec)
postmaster pablo (1994-05-28 17:15:46.66)
              received 57 bytes in 0.634 seconds (89 bytes/sec)
postmaster pablo (1994-05-28 17:15:49.91)
              received 1898 bytes in 1.599 seconds (1186 bytes/sec)
postmaster pablo (1994-05-28 17:15:51.67)
              received 65 bytes in 0.555 seconds (117 bytes/sec)
postmaster pablo (1994-05-28 17:15:55.71)
              received 3217 bytes in 2.254 seconds (1427 bytes/sec)
postmaster pablo (1994-05-28 17:15:57.31)
              received 65 bytes in 0.590 seconds (110 bytes/sec)
```

Opět byly řádky rozděleny tak, aby se vešly na šířku stránky.

Třetím souborem je soubor `Debug`. Do tohoto souboru se zapisují ladicí informace. Pokud používáte ladění, měli byste se ujistit, že má tento soubor nastavena přístupová práva 600. V závislosti na vámi vybraném ladicím režimu může tento soubor obsahovat přihlašovací jméno a heslo, které používáte při spojení se vzdáleným systémem.

Některé binární soubory protokolu UUCP, které jsou součástí distribucí operačního systému Linux, byly sestaveny se zapisováním do log-souborů, které jsou kompatibilní se standardem HDB. Protokol HDB UUCP používá velké množství log-souborů, které jsou uloženy v adresářové struktuře pod adresářem `/var/spool/uucp/.Log`. Tento adresář obsahuje další tři podadresáře, konkrétně `uucico`, `uuxqt` a `uux`. V nich jsou uloženy záznamy ve formě log-souborů, které generuje každý z uvedených programů. Názvy log-souborů se pro každý systém liší. Tedy výstup z programu *uucico* půjde při spojení se systémem **pablo** do souboru `.Log/uucico/pablo`, zatímco výpis z následně spuštěného příkazu `uuxqt` půjde do souboru `.Log/uuxqt/pablo`. Nicméně řádky uvedené v různých log-souborech jsou totožné s řádky, které jsou uvedeny v log-souborech při použití zapisování, které je kompatibilní s protokolem Taylor UUCP.

Když u zapisování do log-souborů, které vyhovují standardu HDB, povolíte ladicí informace, pak tyto ladicí informace půjdou do adresáře `.Admin`, který se nachází pod adresářem `/var/spool/uucp`. Při hovorech směrem ze systému půjdou tyto informace do souboru `.Admin/audit.local`, zatímco v případě, když někdo volá váš systém, půjde výstup z programu `uucico` do souboru `.Admin/audit`.

Kapitola

13

Elektronická pošta

Elektronická pošta představuje jedno z nejvýznamnějších využití síťových služeb od doby, kdy byla síť vynalezena. Původně to byla jednoduchá služba, která kopírovala soubor z jednoho počítače do druhého, kde ho připojila k souboru *poštovní schránky* příjemce. V zásadě tento proces představuje podstatu elektronické pošty, i když stále rostoucí síť vedla se svými složitými směrovacími požadavky a se svým stále se zvyšujícím počtem zpráv k nutnosti vynalezení propracovanějšího schématu.

Byly vynalezeny různé standardy výměny elektronické pošty. Systémy v síti Internet se drží standardu, který byl uveden v dokumentu RFC 822 a který byl doplněn několika dalšími dokumenty RFC. Tyto dokumenty RFC popisovaly způsob přenášení speciálních znaků apod., který není závislý na počítačové platformě. Mnoho nových nápadů bylo doplněno teprve nedávno a vznikla tzv. „multimediální pošta“, kdy mohou být v poštovních zprávách obsaženy i obrázky a zvuky. Další standard definovala společnost CCITT a nazývá se X.400.

Na platformu Unixu byl přenesen poměrně velký počet programů pro přenos pošty. Jedním z nejznámějších programů je `sendmail`. Byl vyvinut na univerzitě v Berkeley a nyní se používá na velkém počtu platforem. Původním autorem tohoto programu je Eric Allman, který v současné době opět aktivně spolupracuje s týmem, který program `sendmail` vytvořil. V Linuxu jsou dostupné dvě implementace programu `sendmail`, jedna z těchto implementací bude popsána v 15. kapitole. V současnosti se vyvíjí verze 8.9.

Nejvíce používaným poštovním agentem v Linuxu je program `smail-3.1.28`, který napsali Curt Landon Noll a Ronald S. Karr. Tento program je součástí většiny distribucí Linuxu. V následujícím výkladu ho budeme zjednodušeně označovat jako `smail`, i když existují i jiné verze tohoto programu, které jsou naprosto odlišné, ale jimi se zde nebudeme zabývat.

Ve srovnání s programem `sendmail` je program `smail` poměrně mladý. Mají-li se oba programy starat o poštu v malém systému, kde neexistují složité směrovací požadavky, jsou jejich možnosti přibližně stejné. Ovšem u velkých systémů vždy vyhrává program `sendmail`, protože má mnohem pružnější konfigurační schéma.

Oba programy `sendmail` a `smail` podporují skupinu konfiguračních souborů, které si musíte přizpůsobit. Kromě informací, které je nutné zadat, aby vůbec mohl poštovní subsystém běžet (například název místního hostitele), lze nastavovat i množství dalších parametrů. Konfiguračnímu souboru programu `sendmail` je zprvu velmi těžké porozumět. Vypadá podobně, jako kdyby si vaše kočka zdřímla na klávesnici, a packu měla položenou na klávese Shift. Konfigurační soubory programu `smail` jsou strukturovanější a lze jim lépe porozumět než konfiguračnímu souboru programu `sendmail`, ale na druhou stranu nabízí uživateli menší volnost při nastavování chování maileru. U malých systémů UUCP nebo u malých systémů v síti Internet je však práce strávená nastavováním obou programů zhruba rovnocenná.

V této kapitole se budeme zabývat pojmem elektronické pošty a jaké problémy budete muset jako správce řešit. Kapitoly 14 a 15 vám poskytnou instrukce, jak poprvé nastavit programy `smail` a `sendmail`. Informace získané v těchto kapitolách by vám měly stačit ke zprovoznění malých systémů, avšak tyto programy nabízejí mnohem více voleb, takže při konfiguraci nejrůznějších vlastností možná strávíte spoustu hodin.

Na konci této kapitoly si ve zkratce probereme nastavení programu `elm`, což je na mnoha unixových systémem, včetně Linuxu, nejčastěji používaný poštovní agent.

Chcete-li získat více informací o problémech, které se týkají elektronické pošty v Linuxu, nahlédněte prosím do dokumentu Vince Skahana Electronic Mail HOWTO, který je pravidelně posílán na adresu **comp.os.linux.announce**.* Zdrojové distribuce programů `elm`, `sendmail` a `smail` také obsahují velmi rozsáhlou dokumentaci, kde byste měli najít odpovědi na většinu otázek týkajících se jejich nastavení. Hledáte-li obecné informace o elektronické poště, existuje spousta dokumentů RFC, které se tímto tématem zabývají. Dokumenty RFC jsou uvedeny v seznamu literatury na konci této knihy.

13.1 Co je to poštovní zpráva?

Poštovní zpráva se zpravidla skládá z textu zprávy, což je text napsaný odesílatelem, a z dat, které označují příjemce, transportní médium atd. a podobají se adrese na dopisní obálce.

Tato administrativní data spadají do dvou kategorií; v první kategorii jsou zastoupena všechna data vztahující se k transportnímu médiu, jako je adresa zasilatele a adresa příjemce. Z to-

* Poznámka korektora: V současné době je primárním zdrojem linuxových zdrojových textů server `ftp://ftp.kernel.org` V České republice je zrcadlo např. na `ftp://ftp.fi.muni.cz/pub/linux/kernel`

hoto důvodu se tato kategorie nazývá *obálka*. V závislosti na tom, kudy prochází daná zpráva, mohou být data z této kategorie transportním softwarem pozměněna.

Druhou kategorii představují všechna data, která jsou nutná pro manipulaci s poštovní zprávou a která se nevztahují k žádnému transportnímu mechanismu. Příkladem může být řádka s předmětem zprávy, seznam všech příjemců nebo datum zaslání dané zprávy. V mnoha sítích se stalo standardem, že tato kategorie dat je uvedena před vlastní poštovní zprávou a tvoří tzv. *poštovní hlavičku*. Od *textu zprávy* je oddělena prázdným řádkem.[1]

Ve světě Unixu používá většina transportních softwarů formát hlavičky, který byl definován v dokumentu RFC 822. Jeho původním záměrem bylo vytvořit standard pro použití v sítích ARPANET, ale protože byl navržen jako nezávislý na prostředí, byl jednoduše upraven pro použití v dalších sítích, včetně mnoha sítí založených na protokolu UUCP.

Nicméně dokument RFC 822 je pouze největším obecným standardem. Vznikly i novější standardy, a to z důvodu zvyšujících se potřeb, jako je šifrování dat, podpora mezinárodní znakové sady a podpora multimediálního rozšíření pošty (MIME).

Ve všech těchto standardech se hlavička zprávy skládá z několika řádek, které jsou odděleny znakem nový řádek. Řádku tvoří název pole, které začíná ve sloupci jedna, a vlastní pole, které je odděleno dvojtečkou a bílým místem. Formát a sémantika každého pole jsou odlišné a závisí na názvu daného pole. Pole hlavičky může pokračovat i na novém řádku v případě, že následující řádek začíná znakem TAB. Pole mohou být uspořádána v libovolném pořadí.

Typická hlavička poštovní zprávy vypadá asi takto:

```
From brewhq.swb.de!ora.com!andyo Wed Apr 13 00:17:03 1994
Return-Path:
Received: from brewhq.swb.de by monad.swb.de with uucp
         (Smail3.1.28.1 #6) id m0pqqlT-00023aB; Wed, 13 Apr 94 00:17
Received: from ora.com (ruby.ora.com) by brewhq.swb.de with smtp
         (Smail3.1.28.1 #28.6) id ; Tue, 12 Apr 94 2
Received: by ruby.ora.com (8.6.8/8.6.4) id RAA26438; Tue, 12 Apr 94
Date: Tue, 12 Apr 1994 15:56:49 -0400
Message-Id:
From: andyo@ora.com (Andy Oram)
To: okir@monad.swb.de
Subject: Re: Your RPC section
```

[1] K poštovní zprávě se obvykle připojuje podpis neboli signatura, jenž obvykle obsahuje informace o autorovi zprávy společně s nějakým vtipem nebo motem. Podpis je oddělen od vlastního textu zprávy řádkem obsahujícím řetězec „--“.

Všechna potřebná pole hlavičky jsou obvykle vygenerována používaným rozhraním maileru, což může být například `elm`, `pine`, `mush` nebo `mailx`. Avšak některá pole jsou volitelná a může je přidat sám uživatel. Například mailer `elm` vám umožní upravit část hlavičky zprávy. Další pole jsou přidána poštovním transportním softwarem. Následuje seznam obecných polí hlavičky a jejich význam:

`From:` Toto pole obsahuje adresu elektronické pošty odesílatele a může eventuálně obsahovat i jeho „skutečné jméno“. Pro toto pole se používá obrovské množství formátů.

`To:` Toto pole obsahuje adresu elektronické pošty příjemce.

`Subject:` Toto pole obsahuje popis obsahu zprávy vyjádřený několika slovy. Pole `Subject` by mělo obsahovat přinejmenším *účel* dané zprávy.

`Date:` Toto pole obsahuje datum, kdy byla zpráva odeslána.

`Reply-to:` Toto pole určuje adresu, na kterou chce odesílatel směrovat odpověď od příjemce. Pole může být užitečné v případě, že máte několik účtů, ale přitom chcete dostávat poštu pouze na adresu, kterou používáte nejčastěji. Toto pole je volitelné.

`Organization:` Toto pole obsahuje společnost, která vlastní počítač a z něhož pochází daná pošta. Máte-li svůj počítač v soukromém vlastnictví, nechte toto pole buď volné, nebo sem zadejte řetězec „private“, případně nějaký nesmysl. Toto pole je volitelné.

`Message-ID:` Toto pole obsahuje řetězec, který vygeneroval transport pošty v původním systému. Vzhledem k této zprávě je to jedinečný řetězec.

`Received:` Toto pole vloží do hlavičky každý systém, který zpracoval vaši poštu (včetně počítačů odesílatele a příjemce). V něm je uveden název daného systému, číslo id zprávy, čas a datum, kdy daný systém obdržel tuto zprávu, dále od kterého systému tato zpráva pochází a který transportní software byl použit k jejímu doručení. Tyto informace se uvádějí z toho důvodu, abyste mohli sledovat směrování pošty a případně si stěžovat příslušné zodpovědné osobě.

`X-`*anything:* Žádný z poštovních programů by si neměl stěžovat na hlavičku, která začíná řetězcem `X-`. Tento řetězec se používá kvůli implementaci doplňkových vlastností, jež zatím nebyly vloženy do standardu RFC, nebo které do něj ani vloženy nebudou. Tento řetězec používá poštovní konference Linux Activists, kde se například pomocí pole hlavičky `X-Mn-Key:` vybírá požadovaný kanál.

Jedinou výjimku z této struktury představuje úplně první řádek. Ten začíná klíčovým slovem `From`, za kterým následuje místo dvojtečky pouze mezera. Aby se toto pole odlišilo od běžného pole `From:`, označuje se často jako pole `From_`. Toto pole obsahuje směrování, kudy musí daná zpráva projít. Směrování je uvedeno pomocí vykřičníkové notace (bude vysvětleno níže), která je charakteristická pro protokol UUCP. Dále obsahuje čas a datum, kdy byla zpráva přijata posledním počítačem, který ji zpracovával, a volitelně může obsahovat i hostitele, ze kterého byla daná zpráva přijata. Protože je toto pole obnovováno každým systémem, který zpracovává danou zprávu, zahrnuje se někdy do dat obálky.

Pole `From_` se uvádí z důvodu zpětné kompatibility s některými staršími mailery, ale jinak se už příliš nepoužívá. Používají ho pouze poštovní rozhraní uživatelů, která na něj spoléhají při označení začátku zprávy v poštovní schránce uživatele. Abyste se vyhnuli potížím s řádkami uvnitř textu zprávy, které také začínají řetězcem „From“, stalo se standardem uvození každého výskytu tohoto řetězce znakem „>“.

13.2 Jakým způsobem se pošta doručuje?

Obecně vytvoříte poštu pomocí nějakého rozhraní maileru, například pomocí programu `mail` nebo `mailx`; nebo pomocí dokonalejších programů, jako je `elm`, `mush` nebo `pine`. Takový program se nazývá *poštovní uživatelský agent* (*mail user agent*), zkráceně MUA. Když se posílá nějaká poštovní zpráva, předá program tuto zprávu ve většině případů rozhraní dalšího programu, který se postará o její doručení. Tento program se nazývá *poštovní přenosový agent* (*mail transport agent*), zkráceně MTA. U některých systémů existují pro místní a vzdálené doručování různí poštovní přenosoví agenti; na dalších systémech existuje pouze jeden agent. Program pro vzdálené doručení se obvykle nazývá `rmail`, dalším obdobným programem je `lmail` (pokud existuje).

Místní doručování znamená samozřejmě více, než jen pouhé přidání příchozí zprávy do poštovní schránky příjemce. Místní agent MTA bude obvykle rozumět schodovitým odkazům (což je nastavení, kdy místní adresy příjemce odkazují na další adresy) a doručování (což je přesměrování pošty uživatele na nějakou jinou destinaci). Také zprávy, které nemohou být doručeny, musí být *odmítnuty*, což znamená, že je systém musí vrátit odesílateli společně s nějakou chybovou zprávou.

U vzdáleného doručování závisí použitý transportní software na povaze daného spojení. Musí-li být pošta doručena po síti na bázi protokolu TCP/IP, použije se protokol SMTP. Zkratka SMTP znamená jednoduchý poštovní přenosový protokol (Simple Mail Transfer Protocol), který je definován v dokumentu RFC 788 a v RFC 821. Protokol SMTP se obvykle přímo spojí s počítačem příjemce a přenos pošty se sjedná s démonem SMTP, který je spuštěn na vzdálené straně.

V sítích na bázi protokolu UUCP nebývá obvykle pošta doručována přímo, ale je doručena požadovanému hostiteli přes několik zprostředkujících systémů. Posílá-li se zpráva pomocí spojení na bázi protokolu UUCP, spustí agent MTA odesílatele obvykle pomocí příkazu `uux` na doručujících systémech program `rmail` a pošle mu na standardní vstup danou zprávu.

Protože se tento proces provádí samostatně pro každou zprávu, může způsobit jak značné pracovní zatížení na hlavním poštovním serveru, tak i přeplnění dočasné fronty protokolu UUCP se stovkami malých souborů, které zabírají neúměrné množství místa na disku.[2] Proto vám někteří agenti MTA umožní seskupit několik zpráv pro vzdálený systém do jednoho dávkového souboru. Dávkový soubor obsahuje příkazy protokolu SMTP, které by za normálních okolností spustil místní hostitel, kdyby se použilo přímé spojení. Tento postup se označuje jako tzv. protokol BSMTP neboli *dávkový* protokol SMTP. Dávka je potom předána programům *tsmtp* nebo `bsmtp` na vzdáleném systému, které zpracují vstup stejným způsobem, jako kdyby došlo k normálnímu spojení.

13.3 Adresy elektronické pošty

U elektronické pošty se adresa skládá přinejmenším z názvu počítače, na kterém je uložena pošta dané osoby, a z identifikace uživatele. Tou může být buď přihlašovací jméno příjemce,, nebo cokoliv jiného. Jiná poštovní adresní schémata, jako je X.400, používají obecnější množinu „atributů“, které slouží k vyhledání hostitele příjemce na adresářovém serveru X.500.

Způsob, jakým je interpretován název počítače, tj. na jakém systému nakonec skončí vaše zpráva a jak se zkombinuje tento název se jménem uživatele příjemce, značně závisí na typu sítě, jíž jste účastníkem.

Systémy v síti Internet dodržují standard popsaný v RFC 822, který vyžaduje notaci **user@host.domain**, kde **host.domain** je plně kvalifikované doménové jméno daného hostitele. Spojovacím členem je znak „zavináč“ (@). Protože tato notace neobsahuje směrování na cílového hostitele, ale (jedinečný) název hostitele, nazývá se tento typ notace *absolutní* adresa.

V původním prostředí protokolu UUCP se běžně používá forma **path!host!user**, kde cesta **path** popisuje pořadí hostitelů, kterými musí zpráva projít, než dorazí k cílovému hostiteli **host**. Tato forma zápisu se nazývá *vykřičníková* notace podle znaku vykřičníku, který se používá v jejím zápisu. Dnes již mnoho sítí založených na protokolu UUCP adoptovalo standard z dokumentu RFC 822 a tudíž bude rozumět i absolutní adrese.

2 To proto, že diskový prostor se obvykle alokuje v blocích o velikosti 1 024 bajtů. Takže i zpráva o velikosti 400 bajtů zabere celý 1 KB.

Tyto dva typy adres se příliš dobře neslučují. Předpokládejme adresu ve tvaru **hostA!user@hostB**. Není zcela jasné, zda znak „@“ bude mít přednost před cestou nebo tomu bude naopak. Pošleme zprávu hostiteli **hostB**, který ji pošle uživateli na adrese **hostA!user,** nebo pošleme zprávu hostiteli **hostA**, který ji doručí uživateli na adrese **user@hostB**?

Adresy, v nichž jsou zastoupeny různé typy operátorů adres, se nazývají *hybridní adresy*. Nejznámější z nich vidíte ve výše uvedeném příkladu. Tato adresa je obvykle vyřešena tak, že operátor „@“ dostane přednost před cestou. Ve výše uvedeném příkladu to znamená, že zpráva bude odeslána nejdříve hostiteli **hostB**.

Existuje však způsob, jak zadat směrování, které by vyhovovalo standardu uvedenému v dokumentu RFC 822: zápis **<@hostA,@hostB:user@hostC>** určuje adresu uživatele **user** na hostiteli **hostC**, kde hostitel **hostC** je dosažitelný přes hostitele **hostA** a **hostB** (v uvedeném pořadí). Tento typ adresy se často označuje jako tzv. *směrovací adresa*.

Dále existuje ještě adresový operátor „%“: U adresy **user%hostB@hostA** bude zpráva nejprve poslána hostiteli **hostA**, který nahradí znak procenta vpravo (pouze v našem případě) znakem „@“. Takže nyní budeme mít adresu **user@hostB** a mailer šťastně doručí vaši zprávu hostiteli **hostB**, který ji doručí uživateli **user**. Tento typ adresy se někdy označuje jako tzv. „Ye Olde ARPANET Kludge“ a jeho používání se v současné době snažíme zabránit. Přesto však tento typ adresy i nadále generuje mnoho poštovních přenosových agentů.

Ostatní sítě stále ještě používají odlišné způsoby adresování. Například sítě na bázi DEC používají jako adresový operátor dvě dvojtečky, takže adresa má tvar **host::user**.[3] A konečně standard X.400 používá zcela odlišné schéma, které popisuje příjemce pomocí skupiny párů atribut-hodnota, například pár země a organizace.

V sítích FidoNet je každý uživatel identifikován kódem, například **2:320/204.9**, který se skládá ze čtyř znaků označujících zónu (2 je pro Evropu), síť (320 označuje Paříž a Banlieue), uzel (což je místní hub) a koncový bod (počítač PC individuálního uživatele). Adresy sítě FidoNet mohou být namapovány na standard definovaný v dokumentu RFC 822; výše uvedená adresa může být zapsána jako **Thomas.Quinot@p9.f204.n320.z2.fidonet.org**. Neříkal jsem náhodou, že názvy domén se dají lehce zapamatovat?

Použití odlišných typů adres má určité důsledky, které si popíšeme dále. Avšak v prostředí, kde platí standard definovaný v dokumentu RFC 822, budete používat jen velmi zřídka nějaké jiné typy adres, než je typ absolutních adres, jako je **user@host.domain**.

[3] Když se snažíte dostat z prostředí, kde platí standard definovaný v dokumentem RFC 822, na adresu v síti DEC, můžete k tomu použít formuli **„host::user“@relay**, kde **relay** je název známé brány mezi sítěmi Internet a DEC.

13.4 Jak pracuje směrování pošty?

Proces doručování zprávy hostiteli příjemce se nazývá *směrování*. Kromě nalezení cesty ze systému odesilatele do cílového systému v sobě tento proces zahrnuje i kontrolu chyb a optimalizaci rychlosti a nákladů.

Existuje obrovský rozdíl ve způsobu, jakým se stará o poštu systém na bázi protokolu UUCP a systém v Internetu. V Internetu provede hlavní práci související se směrováním dat na hostitele příjemce (který je známý pomocí své IP-adresy) síťová hladina IP, zatímco v zóně protokolu UUCP musí být směrování provedeno uživatelem nebo ho musí vygenerovat poštovní přenosový agent.

13.4.1 Směrování pošty v Internetu

V Internetu záleží pouze na cílovém hostiteli, zda se vůbec uskuteční nějaké konkrétní směrování pošty. Implicitně je nastaveno přímé doručení zprávy cílovému hostiteli. To se provede tak, že systém vyhledá jeho IP-adresu a skutečné směrování dat ponechá na transportní hladině IP.

Většina systémů bude obvykle chtít směrovat veškerou příchozí poštu na dostupný poštovní server, který je schopen postarat se o veškerou dopravu pošty a který ji potom předá místním hostitelům. Pro svoji místní doménu oznámí systém tuto službu v databázi systému DNS pomocí tzv. záznamu MX. Zkratka MX znamená *poštovní server* (*Mail Exchanger*) a obvykle je v tomto záznamu uvedeno, že si hostitel serveru přeje pracovat jako doručovatel pošty pro všechny počítače v rámci příslušné domény. Záznamy MX lze také použít při správě pošty určené pro hostitele, kteří nejsou připojeni k Internetu, příkladem budiž sítě na bázi protokolu UUCP nebo hostitelé v sítích společností, kteří obsahují důvěrná data.

Záznamy MX mají přiřazenu tzv. *prioritu*. Priorita je udávána celým číslem. Má-li některý hostitel několik poštovních serverů, pokusí se poštovní přenosový agent přenést zprávu na ten poštovní server, který má nejnižší prioritu, a pouze v případě, že na tomto serveru neuspěje, se bude snažit spojit s hostitelem, který má vyšší prioritu. Je-li místní hostitel zároveň poštovním serverem pro cílovou adresu, nesmí doručit zprávu na žádného hostitele uvedeného v záznamu MX, který má vyšší prioritu než má on sám; toto je bezpečný způsob, jak zabránit vytváření poštovních smyček.

Předpokládejme, že například organizace Foobar Inc. chce veškerou poštu spravovat na svém počítači s názvem **mailhub**. Potom bude mít v databázi DNS přibližně následující záznam MX:

```
foobar.com          IN   MX     5   mailhub.foobar.com
```

Tento záznam říká, že na adrese **mailhub.foobar.com** bude poštovní server pro doménu **foobar.com** s prioritou 5. Hostitel, který chce doručit zprávu na adresu **joe@greenhouse.foobar.com**, vyhledá pomocí systému DNS doménu **foobar.com** a zjistí, že záznam MX ukazuje na hostitele **mailhub**. Pokud neexistuje žádný záznam MX s prioritou menší než 5, bude zpráva doručena poštovnímu serveru **mailhub**, který ji posléze odešle na hostitele **greenhouse**.

Výše uvedený příklad je skutečně pouze náčrt toho, jak pracují záznamy MX. Chcete-li více informací o směrování v síti Internet, nahlédněte prosím do dokumentu RFC 974.

13.4.2 Směrování pošty v sítích na bázi protokolu UUCP

Směrování pošty v sítích na bázi protokolu UUCP je mnohem komplikovanější než v síti Internet, protože vlastní transportní software neprovádí žádné směrování. Dříve musela být veškerá pošta adresována pomocí vykřičníkové notace, která uváděla seznam hostitelů, pomocí kterých se doručovala pošta. Jednotliví hostitelé byli odděleni vykřičníkem, za kterým následovalo jméno uživatele. Pokud jste chtěli adresovat dopis uživateli Janet na počítač s názvem **moria**, museli jste zadat cestu **eek!swim!moria!janet**. Tato formule poslala poštu z vašeho hostitele na hostitele **eek**, z něj pak na hostitele **swim** a nakonec dorazila pošta z hostitele **swim** na hostitele **moria**.

Zcela zřejmá nevýhoda této techniky spočívá v tom, že si musíte pamatovat síťovou topologii, rychlá spojení atd. Ale ještě horší věcí je, že změna v uspořádání síťové topologie – například odstranění některých spojení nebo odstranění nějakého hostitele – může způsobit nedoručení zprávy, protože jste nebyli obeznámeni s provedenými změnami. A konečně v případě, že změníte místo svého působení, budete pravděpodobně muset aktualizovat veškerá směrování.

Z jednoho důvodu bylo nutné používat směrování od zdroje. Tím důvodem byla přítomnost nejednoznačných názvů hostitelů. Předpokládejme, že existují dva systémy s názvem hostitele **moria**, jeden systém se nachází ve Spojených státech amerických a druhý ve Francii. Na který systém ale adresa **moria!janet** odkazuje? To bude jasné pouze až tehdy, uvedete-li cestu, pomocí které se lze spojit s hostitelem **moria**.

Prvním krokem, který vedl k odstraňování nejednoznačných názvů hostitelů, bylo založení *projektu mapování názvů pro protokol UUCP* (*The UUCP Mapping Project*). Tento projekt je umístěn na Rutgers University a registruje všechny oficiální názvy protokolu UUCP společně se sousedy UUCP a s jejich geografickým umístěním. Projekt zajišťuje, aby nebyl žádný název hostitele použit dvakrát. Informace získané projektem mapování názvů jsou zveřejňovány jako tzv. *Usenetové mapy* (*Usenet Maps*), které jsou pravidelně distribuovány pomocí Usenetu.[4] Typická položka systému v mapě vypadá (po odstranění komentářů) následovně:

[4] Mapy systémů registrovaných v rámci projektu mapování názvů pro protokol UUCP jsou distribuovány prostřednictvím diskusní skupiny **comp.mail.maps**; ostatní organizace mohou pro své sítě zveřejňovat samostatné mapy.

```
moria
        bert(DAILY/2)
        swim(WEEKLY)
```

Tato položka uvádí, že hostitel **moria** má spojení s hostitelem **bert**, se kterým se spojuje dvakrát denně. Dále má spojení s hostitelem **swim**, s nímž se spojuje každý týden. K formátu souboru s mapami se dále ještě vrátíme.

Pomocí informací o jednotlivých spojeních, které jsou uvedeny v souborech s mapami, je možné automaticky vygenerovat celou cestu z vašeho hostitele k libovolnému cílovému systému. Tyto informace se obvykle ukládají v souboru `paths`, někdy se tento soubor také nazývá *databáze cest*. Předpokládejme, že v souboru map stojí, že s hostitelem **bert** se můžete spojit přes hostitele **ernie**. V tom případě by položka pro hostitele **moria** v souboru `paths`, který byl vygenerován z části mapy, mohla vypadat asi takto:

```
moria           ernie!bert!moria!%s
```

Pokud nyní zadáte cílovou adresu **janet@moria.uucp**, vybere agent MTA výše uvedené směrování a pošle zprávu na hostitele **ernie** a jako adresu na obálce uvede **bert!moria!janet.**

Není ovšem vhodné vytvářet soubor `paths` ze všech usenetových map. V nich jsou obvykle informace poměrně zkreslené a často i zastaralé. Z tohoto důvodu se vytváří soubor `paths` ze všech světových map protokolu UUCP pouze na několika hlavních hostitelích. Většina systémů spravuje pouze informace o systémech, které jsou v jejich okolí, a poštu pro hostitele, které nenajdou ve své databázi, pošlou chytřejším hostitelům, jež mají kompletnější směrovací informace. Toto schéma se nazývá *směrování na chytřejší hostitele* (*smart-host routing*). Hostitelé, kteří udržují pouze jediné poštovní spojení pomocí protokolu UUCP (takoví hostitelé se též nazývají *koncové systémy* (*leaf sites*)), sami neprovádí žádné směrování; plně se spoléhají na svého chytřejšího hostitele.

13.4.3 Kombinace protokolu UUCP se standardem definovaným v dokumentu RFC 822

Zatím je nejlepším lékem na problémy se směrováním pošty v sítích na bázi protokolu UUCP převzetí systému DNS do sítí UUCP. Samozřejmě, že pomocí protokolu UUCP se nemůžete dotazovat jmenného serveru. Některé systémy UUCP však vytvořily malé domény, které vnitřně koordinují směrování pošty. V mapách tyto domény zveřejní pouze jednoho nebo dva hostitele, kteří budou označení jako poštovní brány. Tím pádem nemusí v mapách existovat položka pro každého hostitele, který se nachází v příslušné doméně. Brány se starají o veškerou poštu, která přichází do domény nebo která z dané domény odchází. Směrovací schéma uvnitř domény je pro okolní svět neviditelné.

Tento princip funguje velmi dobře se směrováním na chytřejší hostitele, které jsme si popsali výše. O globální směrovací informace se starají pouze brány; vedlejší hostitelé příslušné domény vystačí pouze s ručně vytvořeným souborem `paths`, který uvádí směrování uvnitř jejich domény a směrování na poštovní server. Dokonce ani poštovní brány nemusí mít informace o směrování na každého hostitele protokolu UUCP, které ve světě existují. Kromě kompletních informací o směrování uvnitř jimi obsluhované domény potřebují mít ve svých databázích pouze směrování na všechny domény. Například níže uvedená položka cesty bude směrovat veškerou poštu určenou pro systémy v doméně **sub.org** na hostitele **smurf**:

```
.sub.org            swim!smurf!%s
```

Veškerá pošta směřující na adresu **claire@jones.sub.org** bude poslána na hostitele **swim** s adresou **smurf!jones!claire**.

Hierarchické uspořádání prostoru s názvy domén umožňuje poštovním serverům kombinovat přesnější směrování s méně přesným směrováním. Například systém ve Francii může mít přesné směrování pro subdomény v rámci domény **fr**, ale veškerou poštu určenou pro hostitele v doméně **us** bude směrovat na nějaký systém ve Spojených státech amerických. V tomto případě směrování na základě domén (takto se tato technika označuje) značně redukuje velikost směrovacích databází a stejně tak i potřebnou administrativní režii.

Avšak hlavním přínosem použití názvu domén v prostředí sítí UUCP je, že shoda se standardem uvedeným v dokumentu RFC 822 umožňuje mezi sítěmi na bázi protokolu UUCP a sítí Internet jednoduchou průchodnost dat. V dnešní době má spousta domén UUCP spojení s internetovou bránou, která se chová jako jejich chytřejší hostitel. Posílání zpráv po Internetu je rychlejší a směrování informací je mnohem spolehlivější, protože hostitelé v Internetu mohou používat místo usenetových map systém DNS.

Aby mohl být systém na bázi protokolu UUCP dosažitelný z Internetu, uvádí obvykle internetové brány záznam MX pro domény na bázi protokolu UUCP (záznamy MX byly popsány výše). Předpokládejme třeba, že hostitel **moria** patří do domény **orcnet.org**. Hostitel **gcc2.groucho.edu** se chová vůči této doméně jako internetová brána. Z toho důvodu použije hostitel **moria** jako svého chytřejšího hostitele adresu hostitele **gcc2**, takže veškeré zprávy pro cizí domény budou doručovány pomocí Internetu. Na druhou stranu hostitel **gcc2** uvede záznam MX pro doménu **orcnet.org** a tím pádem může doručit veškerou příchozí poštu určenou pro systémy v doméně **orcnet** na hostitele **moria**.

Nyní zůstal už jen poslední problém, totiž že transportní programy protokolu UUCP neumí obsloužit plně kvalifikovaná doménová jména. Většina balíků UUCP byla navržena tak, aby zvládla názvy systémů do délky osmi znaků, některé dokonce i méně, a použití nealfanumerických kláves, jako jsou tečky, je u většiny z nich absolutně nepřípustné.

Proto je nutná existence určitého převodu mezi názvy odpovídajícími standardu, který byl definován v dokumentu RFC 822, a názvy hostitelů protokolu UUCP. Způsob, jakým je tento převod prováděn, zcela závisí na typu implementace. Obecný způsob, jak namapovat názvy FQDN na názvy protokolu UUCP, spočívá v použití mapování přímo v souboru cest:

```
moria.orcnet.org    ernie!bert!moria!%s
```

Tato formule vytvoří z adresy, která udává plně kvalifikovaný název domény, čistokrevnou vykřičníkovou notaci pro použití v protokolu UUCP. Některé mailery k tomuto účelu poskytují speciální soubory; například program `sendmail` používá soubor `uucpxtable`.

Někdy se při posílání pošty ze sítě na bázi protokolu UUCP do Internetu vyžaduje zpětná transformace (která se hovorově označuje jako tzv. přiřazení domény). Pokud odesilatel pošty použije jako cílovou adresu plně kvalifikované doménové jméno, lze se tomuto problému vyhnout tak, že se při doručování zprávy chytřejšímu hostiteli neodstraní název domény z adresy na obálce. Nicméně stále ještě existují systémy, které nejsou součástí žádné domény. I takovéto systémy lze rozčlenit do domén, což se provede tak, že se k nim připojí fiktivní doména **uucp**.

13.5 Formát souborů map a cest

Databáze cest poskytuje v sítích na bázi protokolu UUCP hlavní informace o směrování. Typická položka vypadá asi takto (název systému je od cesty oddělen tabulátory):

```
moria.orcnet.org  ernie!bert!moria!%s
moria              ernie!bert!moria!%s
```

Tento záznam uvádí, že každá zpráva určená pro hostitele **moria** bude doručena přes hostitele **ernie** a **bert**. Je zde zadán jak plně kvalifikovaný název, tak i název pro protokol UUCP. Oba názvy jsou zde uvedeny pro případ, že by mailer neměl samostatný způsob pro mapování mezi těmito dvěma jmennými prostory.

Pokud chcete směrovat všechny zprávy určené pro hostitele v rámci nějaké domény na jejich poštovní brány, můžete v databázi cest zadat také cestu, která bude mít jako cíl název domény, před nímž bude uvedena tečka. Pokud mohou být například hostitelé v doméně **sub.org** dosažitelní přes poštovní bránu **swim!smurf**, mohla by položka v databázi cest vypadat následovně:

```
.sub.org        swim!smurf!%s
```

Vytvoření souboru cest je přijatelné pouze v případě, že provozujete systém, který neobsahuje příliš mnoho informací o směrování. Provádíte-li směrování pro velké množství hostitelů, bude lepší k tomuto účelu použít příkaz `pathalias`, který umí vytvořit soubor cest ze souborů map. Mapy lze spravovat daleko lépe, protože konkrétní systém lze přidat nebo odstranit tak, že v mapě upravíte jeho položku a necháte znovu vytvořit soubor map. I když se mapy zveřejňované usenetovým projektem mapování názvů zatím ke směrování moc nepoužívají, mohou menší sítě na bázi protokolu UUCP poskytovat informace o směrování pomocí svých vlastních skupin map.

Soubor map se skládá především ze seznamu systémů, ve kterém má každý systém uveden seznam systémů, s nimiž se může spojit nebo které se mohou spojit s ním. Název systému začíná ve sloupci jedna a za ním následuje seznam jednotlivých spojení, která jsou vzájemně oddělena čárkami. Seznam může pokračovat i na dalších řádcích v případě, že následující řádek začíná znakem tabulátor. Každý záznam o spojení je tvořen názvem systému, za kterým je v hranatých závorkách uvedena jeho režie. Režie představuje aritmetický výraz složený z čísel a symbolické režie. Řádky, začínající znakem (#), jsou ignorovány.

Jako příklad uvažujme hostitele **moria**, který se spojuje dvakrát denně s hostitelem **swim.twobirds.com** a jedenkrát týdně s hostitelem **bert.sesame.com**. Kromě toho pro spojení s hostitelem **bert** používá pouze pomalý modem s rychlostí 2 400 bps. Hostitel **moria** by měl v souboru map zveřejnit následující položku:

```
moria.orcnet.org
        bert.sesame.com(DAILY/2),
        swim.twobirds.com(WEEKLY+LOW)

moria.orcnet.org = moria
```

Poslední řádek říká, že hostitel **moria** může být rozpoznán i na základě svého názvu pro protokol UUCP. Všimněte si, že musí být uvedena režie *DAILY/2*, protože volání dvakrát denně ve skutečnosti snižuje režii tohoto spojení na polovinu.

Při použití takovýchto souborů map je schopen příkaz `pathalias` vypočítat optimální směrování na libovolný cílový systém, který je uveden v souboru cest. Dále je schopen z těchto souborů vytvořit databázi cest, která může být poté používána pro směrování do těchto systémů.

Příkaz `pathalias` poskytuje množství dalších vlastností, například skrytí systému (tj. že systémy mohou být přístupné pouze pomocí brány) atd. Viz manuálové stránky příkazu `pathalias`, kde naleznete více detailů a dále i kompletní seznam režií jednotlivých spojení.

Komentáře v souboru map obvykle obsahují doplňkové informace o systémech, které jsou v tomto souboru popsány. Tyto komentáře musí odpovídat pevně stanovenému formátu, takže je lze získat z map. Například program `uuwho` používá k zobrazení těchto informací, které jsou mimochodem naformátovány moc hezky, databázi vytvořenou ze souborů map.

Pokud svůj systém zaregistrujete u organizace, která distribuuje svým členům soubory map, budete obvykle muset v mapě vyplnit přibližně takovýto záznam.

Následuje vzorová položka mapy (de facto jde o položku, která popisuje můj systém):

```
#N       monad, monad.swb.de, monad.swb.sub.org
#S       AT 486DX50; Linux 0.99
#O       private
#C       Olaf Kirch
#E       okir@monad.swb.de
#P       Kattreinstr. 38, D-64295 Darmstadt, FRG
#L       49 52 03 N / 08 38 40 E
#U       brewhq
#W       okir@monad.swb.de (Olaf Kirch); Sun Jul 25 16:59:32 MET DST
#
monad    brewhq(DAILY/2)
# Domains
monad = monad.swb.de
monad = monad.swb.sub.org
```

Bílé místo následující za prvními dvěma znaky je znak tabulátoru. Význam většiny polí je zřejmý; detailní popis obdržíte od jakékoliv domény, u které se zaregistrujete. Největší zábavou je rozluštit význam pole *L*: To uvádí vaše geografické umístění ve formátu zeměpisná šířka / zeměpisná délka a používá se při vykreslování postscriptových map, které ukazují všechny systémy v rámci dané země nebo všechny systémy na světě.[5]

13.6 Konfigurace programu elm

Zkratka `elm` znamená „elektronická pošta“ (electronic mail) a tento program je jedním z nejvýstižněji pojmenovaných unixových nástrojů. Program `elm` poskytuje celoobrazovkové rozhraní s propracovanou nápovědou. Nebudeme se zde zabývat způsobem jeho použití, ale zdržíme se pouze u jeho konfiguračních voleb.

5 Tyto mapy jsou pravidelně posílány na adresu **news.lists.ps-maps**. Předem vás varuji, jsou poměrně objemné!

Teoreticky můžete provozovat program `elm` bez jakékoliv konfigurace a přitom bude vše pracovat správně – kéž byste byli šťastní. Existuje však několik voleb, které je potřeba nastavit, i když budou potřeba jen při určitých událostech.

Při startu si program `elm` přečte několik konfiguračních proměnných ze souboru `elm.rc`, který se nachází v adresáři `/usr/lib/elm`. Potom se pokusí přečíst z vašeho domovského adresáře soubor `.elm/elmrc`. Tento soubor si obvykle nebudete vytvářet sami. Program ho vytvoří za vás, když z nabídky options vyberete volbu „save options“.

Sada voleb použitá v souboru `elmrc` je také dostupná v globálním souboru `elm.rc`. Většina nastavení uvedená v soukromém souboru `elmrc` potlačí nastavení uvedená v globálním souboru.

13.6.1 Volby v globálním souboru programu elm

V globálním souboru `elm.rc` musíte nastavit volby, které se týkají názvu vašeho hostitele. Například ve společnosti Virtual Brewery by mohl soubor pro hostitele **vlager** obsahovat následující informace:

```
#
# Jméno hostitele
hostname = vlager
#
# Jméno domény
hostdomain = .vbrew.com
#
# Plně kvalifikované doménové jméno
hostfullname = vlager.vbrew.com
```

Tyto volby poskytují programu `elm` informace o názvu místního hostitele. I když se tyto informace používají jen zřídka, měli byste je mít nastaveny. Poznamenejte si, že tyto informace mají význam pouze v případě, že je uvedete v globálním konfiguračním souboru; nacházejí-li se ve vašem soukromém souboru `elmrc`, budou ignorovány.

13.6.2 Mezinárodní znakové sady

V minulosti byly navrženy doplňky standardu definovaného v dokumentu RFC 822, které by umožnily podporu různých typů zpráv, například prostý text, binární soubory, postscriptové soubory atd. Skupina standardů a dokumentů RFC, které vyhovují těmto aspektům, se obecně označují zkratkou MIME (Multipurpose Internet Mail Extensions). Kromě jiného tyto

standardy umožňují příjemci zjistit, zda byla při psaní zprávy použita jiná znaková sada než standardní znaková sada ASCII, například francouzské akcenty nebo německé přehlásky. V určitých mezích tyto standardy podporuje i program `elm`.

Znaková sada, kterou vnitřně používá operační systém Linux, se obvykle označuje jako ISO-8859-1, což je název standardu, který tato znaková sada splňuje. Tento standard je také známý pod označením Latin-1. Všechny zprávy používající znaky z této znakové sady by měly mít ve své hlavičce následující řádek:

```
Content-Type:  text/plin; charset=iso-8859-1
```

Přijímací systém by měl toto pole rozeznat a při zobrazování zprávy by měl provést příslušná opatření. Pro zprávy typu *text/plain* (prostý text) je implicitně nastaveno pole znakové sady *charset* na hodnotu *us-ascii*.

Aby bylo možné zobrazit zprávy napsané v jiné znakové sadě, než je znaková sada ASCII, musí program `elm` vědět, jak má tyto znaky zobrazit. Pokud program `elm` obdrží zprávu, ve které má pole *charset* jinou hodnotu než *us-ascii* (nebo při zachování stejné znakové sady bude nastaven jiný typ obsahu zprávy než *text/plain*), pokusí se implicitně zobrazit zprávu pomocí příkazu s názvem `metamail`. Zprávy vyžadující pro zobrazení program `metamail` mají na obrazovce přehledu zobrazeno v prvním sloupci písmeno ‘`M`‘.

Protože je implicitní znakovou sadou operačního systému Linux znaková sada ISO-8859-1, není nutné pro zobrazení zpráv vytvořených pomocí této znakové sady používat program `metamail`. Je-li programu `elm` sděleno, že zobrazení odpovídá standardu ISO-8859-1, pak se program `metamail` nepoužije, ale zpráva bude zobrazena přímo. To zajistíte nastavením následující volby v globálním souboru `elm.rc`:

```
displaycharset = iso-8859-1
```

Pamatujte, že tuto volbu byste měli nastavit i v případě, že neplánujete posílání nebo přijímání zpráv, které obsahují jiné znaky než definuje standard ASCII. Tato volba se uvádí proto, že lidé posílající takovýto typ zpráv obvykle nakonfigurují svůj mailer tak, aby vložil do hlavičky pošty patřičné pole `Content-Type:`, a to bez ohledu na skutečnost, zda je či není daná zpráva napsána ve znakové sadě ASCII.

Ale pouze nastavení této volby v souboru `elm.rc` nestačí. Problém spočívá v tom, že při zobrazování zprávy pomocí vestavěného prohlížeče zavolá program `elm` pro každý zobrazovaný znak příslušnou funkci knihovny, aby zjistil, zda daný znak je či není zobrazitelný. Implicitně bude tato funkce považovat za zobrazitelné pouze znaky splňující standard ASCII a všechny ostatní znaky zobrazí jako „`^?`“. Tento problém lze vyřešit tak, že nastavíme pro-

měnnou prostředí *LC_CTYPE* na hodnotu *ISO-8859-1*. Tato proměnná sdělí knihovně, aby považovala za zobrazitelné všechny znaky ze znakové sady Latin-1. Podpora této i dalších vlastností je dostupná od verze knihovny *libc-4.5.8*.

Při posílání zpráv, které obsahují speciální znaky ze znakové sady ISO-8859-1, byste se měli ujistit, že jste v souboru `elm.rc` nastavili ještě další dvě proměnné:

```
charset = iso-8859-1
textencoding = 8bit
```

Tyto volby způsobí, že program `elm` uvede v hlavičce pošty, že příslušná zpráva byla vytvořena pomocí znakové sady ISO-8859-1 a že bude poslána ve formě 8bitových hodnot (implicitně se veškeré znaky překládají na 7bitové hodnoty).

Samozřejmě, že jakoukoliv z těchto voleb lze místo v globálním konfiguračním souboru uvést v soukromém souboru `elmrc`.

Kapitola

14

Připravení a spuštění programu smail

Tato kapitola je rychlým úvodem do problematiky nastavení programu `smail` a obsahuje také přehled funkcí, které tento program nabízí. I když je program `smail` ve svém chování značně kompatibilní s programem `sendmail`, jsou jejich konfigurační soubory naprosto odlišné.

Hlavním konfiguračním souborem je `/usr/lib/smail/config`. Tento soubor musíte vždy upravit tak, aby obsahoval hodnoty charakteristické pro váš systém. Je-li váš systém pouze koncovým systémem protokolu UUCP, pak nebudete mít moc práce. Ke konfiguraci směrování a transportních voleb lze použít i další soubory, o nichž se také krátce zmíníme.

Program `smail` implicitně ihned zpracuje a doručí veškerou příchozí poštu. Jestliže máte poměrně značné dopravní zatížení, můžete sdělit programu `smail`, aby seskupoval veškeré zprávy do tzv. *fronty*, kterou bude potom zpracovávat pouze v pravidelných intervalech.

Spravujete-li poštu v prostředí sítě na bázi protokolu TCP/IP, běží často program `smail` v režimu démona: při zavádění systému je spuštěn ze skriptu `rc.inet2` a přesune se na pozadí, kde čeká na příchozí spojení TCP na portu pro protokol SMTP (což je obvykle port číslo 25). To je velmi prospěšné v případě, kdy očekáváte značné množství pošty, protože program `smail` se nebude zvlášť spouštět při každém příchozím spojení. Alternativní možnost představuje přenechání správy portu pro protokol SMTP super serveru `inetd`, který při jakémkoliv výskytu příchozího spojení spustí program `smail`.

Program `smail` obsahuje množství příznaků, které ovlivňují jeho chování; kdybychom je zde všechny detailně popsali, stejně by vám to asi moc nepomohlo. Naštěstí program `smail` podporuje množství standardních operačních režimů, které jsou povoleny, pokud ho vyvoláte pomocí speciálního názvu příkazu, například pomocí příkazu `rmail` nebo `smtpd`. Tyto přezdívky obvykle představují symbolické odkazy na vlastní binární kód programu `smail`. S většinou z nich se setkáme, když se budeme bavit o různých vlastnostech programu `smail`.

V každém případě by měly existovat dva symbolické odkazy na program `smail`; konkrétně jsou to příkazy `/usr/bin/rmail` a `/usr/sbin/sendmail`.[1] Když sestavíte a pošlete poštovní zprávu pomocí uživatelského agenta, například pomocí programu `elm`, bude zpráva předána programu `rmail`, který ji doručí. Seznam příjemců je předáván na příkazové řádce. To samé se děje s příchozí poštou při použití protokolu UUCP. Avšak některé verze programu `elm` spustí místo programu `rmail` program `/usr/sbin/sendmail`, takže budete potřebovat oba tyto programy. Máte-li například uložen program `smail` v adresáři `/usr/local/bin`, napište na příkazovém řádku následující příkazy:

```
# ln -s /usr/local/bin/smail /usr/bin/rmail
# ln -s /usr/local/bin/smail /usr/sbin/sendmail
```

Chcete-li se při konfiguraci programu `smail` pouštět do složitějších úkolů, nahlédněte do manuálových stránek programů `smail`*(1)* a `smail`*(5)*. Pokud nejsou tyto informace součástí vaší oblíbené distribuce operačního systému Linux, můžete je získat ze zdrojového kódu programu `smail`.

14.1 Nastavení protokolu UUCP

V případě použití programu `smail` pouze v prostředí sítí na bázi protokolu UUCP je základní instalace poměrně jednoduchá. Nejprve se musíte ujistit, že máte dva symbolické odkazy na výše zmíněné příkazy `rmail` a `sendmail`. Očekáváte-li z ostatních systémů příjem dávkového protokolu SMTP, musíte mít navíc spojení mezi příkazem `rsmtp` a programem `smail`.

V distribuci programu `smail` od Vince Skahana najdete vzorový konfigurační soubor. Jeho název je `config.sample` a je umístěn v adresáři `/usr/lib/smail`. Musíte ho zkopírovat do souboru s názvem *config* a upravit ho tak, aby obsahoval hodnoty charakteristické pro váš systém.

Předpokládejme, že váš systém se nazývá **swim.twobirds.com** a je registrován v mapě protokolu UUCP pod názvem **swim**. Váš chytřejší (smart) hostitel je hostitel **ulysses**. V takovém případě by měl váš soubor `config` vypadat asi takto:

```
#
# Naše doménová jména
visible domain=two.birds:uucp
#
# Naše jméno pro odchozí poštu
```

[1] Toto je nové standardní umístění programu `sendmail`, které vyhovuje standardu linuxového souborového systému (Linux File System Standard). Další běžné umístění je v adresáři `/usr/lib`.

```
visible name=swim.twobirds.com
#
# UUCP jméno
uucp name=swim.twobirds.com
#
# Náš chytřejší hostitel
smart host=ulysses
```

První příkaz sděluje programu `smail` domény, ke kterým patří váš systém. Zde zadejte názvy domén, které budou navzájem odděleny dvojtečkami. Je-li název vašeho systému registrován v mapě protokolu UUCP, měli byste sem přidat i doménu **uucp**. V případě výskytu poštovní zprávy si program `smail` zjistí pomocí systémového volání *hostname(2)* název vašeho hostitele, pak porovná adresu příjemce s názvem hostitele, přitom bude postupně zkoušet všechny názvy uvedené v tomto seznamu. Jestliže se bude adresa shodovat s některým z těchto názvů nebo s některým z nekvalifikovaných názvů hostitelů, bude považovat příjemce za místního a program `smail` se pokusí doručit zprávu místnímu hostiteli. V opačném případě bude příjemce považován za vzdáleného a program `smail` se pokusí doručit zprávu cílovému hostiteli.

Volba *visible_name* by měla uvádět jeden plně kvalifikovaný název domény vašeho systému, který chcete použít u odcházející pošty. Tento název je používán pro veškerou odcházející poštu při generování adresy zasilatele. Musíte se ujistit, že používáte název, u kterého program `smail` pozná, že odkazuje na místního hostitele (tj. na název hostitele uvnitř některé z domén, které jsou vypsány ve volbě *visible_domain*). V opačném případě by odpovědi na vaši poštu nenašly správnou cestu k vašemu systému.

Poslední příkaz nastavuje cestu, která se používá při směrování na chytřejšího hostitele (bylo popsáno ve stati 13.4). U tohoto vzorového nastavení doručí program `smail` chytřejšímu hostiteli veškerou poštu adresovanou na vzdáleného hostitele. Cesta zadaná ve volbě *smart_path* bude použitá pro směrování na chytřejšího hostitele. Protože budou zprávy doručeny pomocí protokolu UUCP, musí volba udávat systém, který zná váš software UUCP. Viz 12. kapitola, kde najdete informace o tom, jak zveřejnit název systému protokolu UUCP.

Ve výše uvedeném souboru se vyskytuje ještě jedna volba, kterou jsme zatím nevysvětlili; tou je *uucp_name*. Důvod pro použití této volby je následující: Program `smail` implicitně používá hodnotu vrácenou příkazem *hostname(2)*. Příkladem může být například návratová cesta, která je v hlavičce zprávy uvedena na řádku *From_*. Pokud váš název hostitele není zaregistrován pomocí projektu mapování názvů pro protokol UUCP, měli byste sdělit programu *smail*, aby místo tohoto názvu hostitele používal vaše plně kvalifikované doménové jméno.[2] To lze provést pomocí volby *uucp_name*, kterou přidáte do souboru `config`.

V adresáři `/usr/lib/smail` se nachází ještě další soubor s názvem `paths.sample`. Tento soubor uvádí vzor, jak by mohl vypadat soubor `paths`. Avšak tento soubor budete potřebovat pouze v případě, že máte poštovní spojení s více než jedním systémem. Je-li to váš případ, musíte si tento soubor sami napsat nebo ho musíte vygenerovat z usenetových map. Soubor `paths` bude popsán dále v kapitole.

14.2 Nastavení pro lokální síť typu LAN

Pokud provozujete systém, který je spojen se dvěma nebo více hostiteli pomocí lokální sítě typu LAN, musíte určit jednoho hostitele, který se bude starat o spojení s okolním světem na bázi protokolu UUCP. Mezi hostiteli uvnitř vaší sítě lokální sítě LAN si budete asi chtít s největší pravděpodobností vyměňovat poštu pomocí protokolu SMTP, který bude pracovat v síti na bázi protokolu TCP/IP. Předpokládejme, že jsme zpět ve společnosti Virtual Brewery a hostitel **vstout** je nastaven jako brána pro protokol UUCP.

V síťovém prostředí je nejlepší uchovávat veškeré uživatelské poštovní schránky v jediném souborovém systému, který mají pomocí systému NFS připojen všichni ostatní hostitelé. To umožňuje uživatelům přesuny z jednoho počítače na druhý, aniž by museli přenášet svou poštu sem a tam (nebo v horším případě museli každé ráno kontrolovat tři nebo čtyři počítače, zda jim nepřišla nějaká nová pošta). Proto budete také požadovat, aby adresa odesilatele byla nezávislá na počítači, na kterém byla pošta napsána. Obvyklou praxí je použít v adrese odesilatele místo názvu hostitele pouze vlastní název domény. Například uživatel Janet by měl zadat místo adresy **janet@vale.vbrew.com** adresu **janet@vbrew.com** Dále si vysvětlíme, jak server pozná, že název domény je korektní název pro váš systém.

Další způsob, jak uchovávat všechny poštovní schránky na centrálním hostiteli, spočívá v použití protokolu POP nebo protokolu IMAP. Protokol POP znamená *poštovní protokol* (*Post Office Protocol*) a umožňuje uživatelům přistupovat ke svým schránkám pomocí jednoduchého

[2] Důvod je následující: Předpokládejme, že název vašeho hostitele je **monad**, avšak tento název není zaregistrován v mapách. V mapách je však zaregistrován nějaký jiný systém **monad**, takže veškerá pošta na adresu **monad!root**, kterou může dokonce poslat i váš přímý soused z hlediska protokolu UUCP, bude doručena druhému systému **monad**. To je rozhodně nepříjemné pro všechny.

spojení na bázi protokolu TCP/IP. Protokol IMAP znamená *interaktivní protokol pro přístup k poště* (*Interactive Mail Access Protocol*) a je podobný protokolu POP, ale trošku obecnější. Na platformu operačního systému Linux byly přeneseny jak servery pro protokol IMAP, tak i servery pro protokol POP. Servery jsou dostupné na adrese **sunsite.unc.edu** v adresáři `/pub/Linux/system/Network`.

14.2.1 Zápis konfiguračních souborů

Konfigurace pro společnost pivovaru pracuje následovně: všichni hostitelé kromě vlastního poštovního serveru s názvem **vstout** směrují veškerou odcházející poštu na poštovní server, k tomu použijí směrování na chytřejšího hostitele. Samotný poštovní server s názvem **vstout** posílá veškerou odcházející poštu na skutečného chytřejšího hostitele, který směruje veškerou poštu ze společnosti pivovaru; tento hostitel se nazývá **moria**.

Standardní soubor `config` určený pro všechny hostitele kromě poštovního serveru s názvem **vstout** vypadá následovně:

```
#
# Naše doména
visible domain=vbrew.com
#
# Naše jméno pro odchozí poštu
visible name=vbrew.com
#
# Cesta k chytřejšímu hostiteli: pomocí SMTP na vstout
smart path=vstout
smart transport=smtp
```

Je to velmi podobné způsobu, který jsme používali u systémů, jež byly připojeny pouze pomocí protokolu UUCP. Hlavní odlišnost spočívá v tom, že transportem používaným k posílání pošty na chytřejšího hostitele je samozřejmě protokol SMTP. Volba *visible_domain* sdělí programu `smail`, aby pro veškerou odcházející poštu používal místo názvu místního hostitele název domény.

Na poštovní bráně pro protokol UUCP s názvem **vstout** vypadá soubor `config` trochu jinak:

```
#
# Naše doména
hostnames=vbrew.com:vstout.vbrew.com:vstout
#
```

```
# Naše jméno pro odchozí poštu
visible name=vbrew.com
#
# Jméno pro UUCP
uucp name=vbrew.com
#
# Cesta k chytřejšímu hostiteli: pomocí UUCP na moria
smart path=moria
smart transport=uux
#
# Spravujeme poštu pro naši doménu
auth domains=vbrew.com
```


Tento soubor `config` používá ke sdělování informací programu `smail` o způsobu volání místního hostitele odlišné schéma. Místo toho, aby mu byl předal seznam domén a aby si nechal vyhledat název hostitele pomocí systémového volání, zadává seznam explicitně. Výše uvedený seznam obsahuje jak plně kvalifikované názvy hostitelů, tak i nekvalifikované názvy hostitelů a navíc obsahuje i samotný název domény. To umožní programu `smail` rozpoznat adresu **janet@vbrew.com** jako místní adresu a doručit zprávu uživateli **janet**.

Volba *auth_domains* uvádí domény, pro něž je směrodatný poštovní server **vstout**. To znamená, že když program `smail` obdrží jakoukoliv adresu směrovanou na adresu `host`.**vbrew.com**, kde hostitel `host` neodpovídá žádnému z místních počítačů, měl by ji odmítnout a vrátit zpět odesilateli. Není-li tato položka uvedena, bude každá taková zpráva poslána chytřejšímu hostiteli, který ji vrátí zpět poštovnímu serveru **vstout,** a tak to pude pokračovat, dokud nebude zlikvidována z důvodu překročení maximálního počtu skoků.

14.2.2 Spuštění programu smail

Nejprve se musíte rozhodnout, zda chcete spouštět program `smail` jako samostatného démona nebo zda necháte spravovat port protokolu SMTP super serverem `inetd`, který při jakémkoliv požadavku ze strany klienta na spojení pomocí protokolu SMTP spustí program `smail`. U poštovního serveru obvykle dáte přednost práci v režimu démona, protože to bude zatěžovat počítač mnohem méně, než stále opakované spouštění programu `smail`, kdykoliv se vyskytne nějaké spojení. Protože poštovní servery doručují také většinu příchozí pošty přímo uživatelům, zvolíte na většině hostitelů práci v režimu super serveru `inetd`.

Ať už zvolíte kterýkoliv z těchto režimů, musíte se ujistit, že máte v souboru `/etc/services` následující položku:

```
smtp            25/tcp          # Simple Mail Transfer Protocol
```

Tato položka definuje číslo portu pro protokol TCP, které by měl použít program `smail` při spojeních pomocí protokolu SMTP. V dokumentu RFC Assigned Numbers je definována implicitní hodnota 25.

Když je program `smail` spuštěn v režimu démona, přejde do pozadí a bude čekat na nějaké spojení na portu pro protokol SMTP. Když k tomuto spojení dojde, spustí druhý proces a bude řídit komunikaci, která probíhá na bázi protokolu SMTP. Démon programu `smail` se většinou spouští pomocí následujícího příkazu ze skriptu `rc.inet2`:

```
/usr/local/bin/smail -bd -q15m
```

Příznak `-bd` zapíná režim démona a příznak `-q15m` sdělí programu `smail`, aby každých 15 minut zpracovával zprávy, které se mezitím nahromadily ve frontě.

Chcete-li k tomuto účelu použít raději super-server `inetd`, měl by váš soubor `/etc/inetd.conf` obsahovat následující řádek:

```
smtp     stream  tcp nowait  root  /usr/sbin/smtpd smtpd
```

Příkaz `smtpd` by měl být symbolickým odkazem na binární soubor programu `smail`. Pamatujte, že po provedení těchto změn musíte donutit super-server *inetd*, aby znovu načetl konfigurační soubor `inetd.conf`. To provedete tak, že mu pošlete signál *HUP*.

Jak režim démona, tak i režim super-serveru `inetd` jsou režimy výhradní. Jestliže máte spuštěný program `smail` v režimu démona, musíte se ujistit, že jste v souboru `inetd.conf` zakázali veškeré řádky, které se vztahují ke službě `smtp`. Podobně se v případě, že necháte super-server `inetd` spravovat program `smail`, musíte ujistit, že ze skriptu `rc.inet2` nespouštíte démona `smail`.

14.3 Když se vám nepodaří prorazit...

Pokud se při instalaci v něčem zmýlíte, máte k dispozici spoustu vlastností, jež vám mohou pomoci nalézt jádro problému. V prvé řadě musíte zkontrolovat všechny log-soubory programu `smail`. Tyto soubory jsou uloženy v adresáři `/var/spool/smail/log` a jejich názvy jsou `logfile`, resp. `paniclog`. Soubor `logfile` vypisuje všechny provedené úkony, zatímco soubor `paniclog` je určen pouze pro výpis chybových zpráv, které se vztahují k chybám konfigurace atp.

Typický záznam v souboru `logfile` vypadá následovně:

```
04/24/94 07:12:04: [m0puwU8-00023UB] received
|             from: root
```

```
|          program: sendmail
|             size: 1468 bytes
 04/24/94 07:12:04: [m0puwU8-00023UB] delivered
|              via: vstout.vbrew.com
|               to: root@vstout.vbrew.com
|          orig-to: root@vstout.vbrew.com
|           router: smart host
|        transport: smtp
```

Tento výpis ukazuje, že zpráva od uživatele **root** směrovaná na **root@vstout.vbrew.com**, byla protokolem SMTP v pořádku doručena na hostitele **vstout**.

U zpráv, které nemůže program `smail` doručit, se vygeneruje v souboru log podobný záznam, jen místo klíčového slova `delivered` v něm bude uvedena chybová zpráva:

```
04/24/94 07:12:04: [m0puwU8-00023UB] received
|             from: root
|          program: sendmail
|             size: 1468 bytes
04/24/94 07:12:04: [m0puwU8-00023UB] root@vstout.vbrew.com ... defer
 (ERR 148) transport smtp: connect: Connection refused
```

Výše uvedená chyba je typická pro situaci, kdy program `smail` správně rozpozná, že má zprávu doručit hostiteli **vstout**, avšak nemůže se spojit se službou SMTP na hostiteli **vstout**. V takovém případě je buď chyba v konfiguraci, anebo ve vašich binárních souborech programu `smail` schází podpora protokolu TCP.

Tento problém není zase tak ojedinělý, jak se mohlo na první pohled zdát. Existují předkompilované binární soubory `smail`, dokonce i v některých distribucích operačního systému Linux, které nemají podporu pro sítě na bázi protokolu TCP/IP. Je-li to právě váš případ, musíte si sami zkompilovat program `smail`. Máte-li nainstalován program `smail`, pak si na něm můžete otestovat, zda podporuje sítě na bázi protokolu TCP. Stačí jen spustit program `telnet` na portu, který je určen pro protokol SMTP. Úspěšné připojení k serveru SMTP je ukázáno níže (vaše vstupy jsou zobrazeny *tímto stylem*):

```
$ telnet localhost smtp
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 monad.swb.de Smail3.1.28.1 #6 ready at Sun, 23 Jan 94
```

```
19:26 MET
QUIT
221 monad.swb.de closing connection
```

Pokud tento test nezobrazí hlavičku protokolu SMTP (to je ten řádek, co začíná hodnotou 220), pak se předtím, než začnete sestavovat program `smail`, ujistěte, že je vaše konfigurace *skutečně* v pořádku. Kompilace vlastního programu `smail` bude popsána dále.

Narazíte-li u programu `smail` na problém, který nepůjde lokalizovat z chybové zprávy vygenerované programem `smail`, můžete vyzkoušet povolit ladicí informace. To lze provést pomocí volby `-d`, za kterou může následovat číslo určující úroveň rozsahu výpisů (mezi volbou a číselným argumentem nesmí být mezera). V takovém případě vypíše program `smail` na obrazovce zprávu o své činnosti, která vám může pomoci při hledání příslušného problému.

[**No, já nevím … Snad to lidé nebudou považovat za příliš směšné:**] Pokud nic nepomůže, pak zkuste spustit program `smail` pomocí volby `-bR` v režimu „darebáka“. Tato volba se uvádí na příkazové řádce. Na manuálové stránce se o této volbě píše následující: „Vstupte do nepřátelské domény, kterou tvoří obrovské poštovní zprávy a svitky standardů ve formě RFC. Pokuste se ji zmenšit na protokolovou úroveň 26 a potom ji zase vraťte zpět.“ I když tato volba vaše problémy nevyřeší, může vás trochu povzbudit a poskytnout vám alespoň malou útěchu.[3]

14.3.1 Sestavení programu smail

Pokud s jistotou víte, že program `smail` nemá zabudovanou podporu pro sítě na bázi protokolu TCP, musíte si obstarat jeho zdrojový kód. Jestliže jste vaši distribuci operačního systému Linux dostali na CD-ROM, pak bude zdrojový kód programu `smail` asi její součástí, v opačném případě ho můžete získat ze sítě pomocí služby FTP.[4]

Při sestavování programu `smail` uděláte nejlépe, když začnete se skupinou konfiguračních souborů od Vince Skahana, které jsou součástí distribuce balíku `newspak`. Abyste ho zkompilovali s podporou síťového ovladače protokolu TCP, musíte nastavit v souboru `conf/EDITME` makro *DRIVER_CONFIGURATION* buď na hodnotu *bsd-network*, nebo na hodnotu *arpa-network*. Hodnota *bsd-network* je vhodná pro instalace v lokálních sítích typu LAN, zatímco instalace v Internetu vyžadují hodnotu *arpa-network*. Rozdíl v těchto dvou hodnotách spočívá v tom, že hodnota *arpa-network* přidá speciální ovladač pro službu BIND, který je schopen rozpoznat záznamy typu MX. Při zadání volby *bsd-network* nebudou tyto záznamy rozpoznány.

3 Pokud máte skutečně špatnou náladu, pak tuto volbu nezkoušejte.

4 Pokud jste si ho koupili od vašeho dodavatele společně s distribucí Linuxu, máte podle licenčních podmínek programu `smail` právo na tento zdrojový kód za cenu „nepatrného poštovního poplatku“.

14.4 Režimy doručování pošty

Jak jsme se již zmínili, program `smail` je schopen okamžitě doručovat zprávy nebo je může ukládat do fronty, kterou zpracuje později. Pokud si zvolíte ukládání zpráv do fronty, bude program `smail` ukládat veškerou poštu do podadresáře `messages`, který je umístěn v adresáři `/var/spool/smail`. Tyto zprávy nebude zpracovávat, dokud mu to explicitně nepřikážete (tento proces se označuje jako tzv. „zpracování fronty“).

Pomocí volby *delivery_mode* umístěné v souboru `config` si můžete vybrat jeden ze tří doručovacích režimů, a to buď režim *foreground*, *background* nebo *queued*. Tyto režimy zvolí doručování na popředí (příchozí zprávy se zpracují okamžitě), na pozadí (zprávy budou doručeny potomkem přijímacího procesu, rodičovský proces skončí okamžitě po vyvolání potomka přijímacího procesu) nebo zvolí doručování pomocí fronty. Pokud je v souboru `config` nastavena booleovská proměnná *queue_only*, bude příchozí pošta vždy posílána do fronty, a to bez ohledu na nastavení volby *delivery_mode*.

Pokud zapnete posílání do fronty, musíte se ujistit, že tato fronta je pravidelně kontrolována; například každých 10 nebo 15 minut. Spouštíte-li program `smail` v režimu démona, musíte na příkazovou řádku přidat volbu `-q10m`, aby se fronta zpracovávala každých 10 minut. Další možností je v těchto intervalech spouštět příkaz `runq` z programu `cron`. `runq` by měl být odkaz na programu `smail`.

Aktuální poštovní frontu si můžete zobrazit spuštěním programu `smail` s volbou `-bp`. Další možností je vytvořit odkaz `mailq` na program `smail`; pak stačí volat příkaz `mailq`:

```
$ mailq -v
m0pvB1r-00023UB From: root  (in /var/spool/smail/input)
                Date: Sun, 24 Apr 94 07:12 MET DST
                Args: -oem -oMP sendmail root@vstout.vbrew.com
Log of transactions:
Xdefer:  reason: (ERR 148) transport smtp:
connect: Connection refused
```

Tento příkaz ukazuje jednotlivé zprávy, které jsou umístěny v poštovní frontě. Záznam provedených transakcí (které jsou zobrazeny pouze v případě, že spustíte příkaz `mailq` s volbou `-v`) vám může poskytnout dodatečný důvod, proč daná zpráva stále čeká na doručení. Pokud zatím nebyl učiněn žádný pokus o doručení zprávy, nebude zobrazen žádný záznam provedených transakcí.

Dokonce i v případě, že nechcete používat poštovní frontu, vloží občas program *smail* zprávy do fronty, pokud zjistí, že okamžité doručení neuspělo z přenosových důvodů. U spojení pomocí protokolu SMTP může být například tímto důvodem nedosažitelnost požadovaného hostitele; avšak zprávy se mohou zpozdit i v případě, kdy se zjistí, že je souborový systém téměř zaplněn. Proto byste měli nechat frontu zpracovávat přibližně každou hodinu (pomocí příkazu `runq`). V opačném případě by veškeré odložené zprávy zůstaly ve frontě navždy viset.

14.5 Různé volby v souboru config

Existuje poměrně velký počet voleb, které lze zadat v konfiguračním souboru `config`. Ačkoliv jsou tyto volby užitečné, nejsou pro správnou funkci programu `smail` nezbytné, a proto se zde jimi nebudeme zabývat. Místo toho se zde zmíníme pouze o několika z nich, které by se vám mohly hodit:

error_copy_postmaster
: Je-li tato booleovská proměnná nastavena, pak bude mít jakákoliv chyba za následek vygenerování zprávy pro účet postmaster. Obvykle se to provádí pouze u chyb, ke kterým došlo z důvodu chybné konfigurace. Tuto proměnnou lze zapnout v souboru `config` tak, že před její název vložíte symbol plus (+).

max_hop_count
: Je-li počet skoků (hops) u dané zprávy (tj. počet hostitelů, jimiž daná zpráva prošla) rovný nebo vyšší než tato hodnota, bude výsledkem pokusů o vzdálené doručení chybová zpráva, která bude doručena odesilateli. Tato volba se používá k tomu, aby se zprávy nemohly dostat do nekonečné smyčky. Počet skoků se obvykle vypočítá z počtu polí *Received:*, které se nacházejí v hlavičce zprávy. Může však být také zadán ručně pomocí volby `-h` zadané na příkazové řádce.

  Tato proměnná je implicitně nastavena na hodnotu 20.

postmaster
: Adresa postmastera. Nemůže-li být adresa **Postmaster** nalezena jako korektní místní adresa, pak se tato volba použije jako poslední možnost. Implicitně je nastavena na hodnotu **root**.

14.6 Směrování a doručování zpráv

Program `smail` člení proces doručení na tři odlišné úkoly, směrovač, řídící modul a transportní modul.

Modul směrovače rozkládá veškeré vzdálené adresy a určuje, ke kterému hostiteli by měla být zpráva dále poslána a který transport se k tomu musí použít. V závislosti na povaze daného spojení mohou být použity rozdílné transporty, například jako transport může být použit protokol UUCP nebo protokol SMTP.

Místní adresy jsou předány řídicímu modulu, který rozloží veškerá směrování nebo veškeré používané přezdívky. Adresou může být například přezdívka nebo poštovní konference, případně může chtít uživatel směrovat svoji poštu na jinou adresu. Pokud je výsledná adresa vzdálená, bude předána modulu směrovače, který zjistí doplňkové směrování, v opačném případě bude podstoupena transportnímu modulu, který ji doručí místnímu hostiteli. Zdaleka nejběžnějším způsobem bude doručování do poštovní schránky, avšak zprávy mohou být také předány nějakému příkazu nebo připojeny k libovolnému souboru.

A konečně transportní modul je zodpovědný za libovolnou vybranou metodu doručení. Tento modul se pokusí doručit poštu a v případě neúspěchu buď vygeneruje skokovou zprávu, nebo odloží danou zprávu pro případné další pokusy.

Program `smail` vám poskytuje při konfiguraci těchto úkolů značnou volnost. Pro každý z těchto úkolů je dostupných několik ovladačů, z nichž si můžete vybrat pouze ty, které budete potřebovat. Tyto ovladače popíšete programu `smail` v několika souborech, konkrétně jsou to soubory `routers`, `directors` a `transports`, které se nacházejí v adresáři `/usr/lib/smail`. Pokud tyto soubory neexistují, budou se předpokládat rozumné implicitní hodnoty, které by měly být vhodné pro spoustu systémů používajících jako transportní modul protokol SMTP nebo protokol UUCP. Chcete-li změnit směrovací pravidla programu `smail` nebo chcete-li upravit používaný transportní modul, pak byste si měli obstarat vzorové soubory ze zdrojové distribuce,[5] zkopírovat tyto soubory do adresáře `/usr/lib/smail` a upravit je tak, aby vyhovovaly vašim potřebám. Vzorové konfigurační soubory jsou uvedeny v dodatku B.

14.7 Směrování zpráv

Když je zachycena zpráva, program `smail` nejprve zkontroluje, zda je cílovým hostitelem místní hostitel nebo vzdálený systém. Je-li cílovým hostitelem adresa jednoho z místních názvů hostitelů nakonfigurovaných v souboru `config`, bude zpráva předána řídicímu modulu. V opačném případě předá program `smail` cílovou adresu několika ovladačům směrovačů, aby zjistil, jakému hostiteli má být daná zpráva poslána. Ovladače směrovačů mohou být popsány v souboru `routers`; pokud tento soubor neexistuje, budou použity implicitní směrovače.

5 Implicitní konfigurační soubory mohou být umístěny v podadresáři `samples/generic` adresáře se zdrojovým kódem.

Cílový hostitel je postupně předán všem směrovačům a zvolí se ten směrovač, který nalezne nejpřesnější směrování. Uvažme zprávu poslanou na adresu **joe@foo.bar.com**. Pak jeden ze směrovačů může znát implicitní směr na všechny hostitele v doméně **bar.com**, zatímco nějaký jiný směrovač může mít informace přímo o vlastním hostiteli **foo.bar.com**. Protože naposledy uvedený směrovač obsahuje přesnější směrování, bude mít přednost před dříve uvedeným směrovačem. Pokud existují dva směrovače, kteří poskytnou „nejlepší směrování“, bude vybrán ten z nich, který je uveden v souboru `routers` dříve.

Nyní tento směrovač určí typ použitého přenosu, například protokol UUCP, a vygeneruje novou cílovou adresu. Ta je pak předána transportnímu modulu společně s názvem hostitele, kterému by se měla zpráva poslat. Ve výše uvedeném příkladu by mohl program `smail` zjistit, že hostitel **foo.bar.com** je dosažitelný pomocí protokolu UUCP, pokud použije cestu **ernie!bert**. Potom program `smail` vygeneruje novou cílovou adresu **bert!foo.bar.com!user** a sdělí transportnímu modulu pro protokol UUCP, aby tuto adresu předal hostiteli **ernie** jako tzv. adresu na obálce.

Při použití implicitního nastavení jsou k dispozici následující směrovače:

- Může-li být adresa cílového hostitele resolvována pomocí volání funkce knihovny *gethostbyname(3)* nebo *gethostbyaddr*(3), pak bude zpráva doručena pomocí protokolu SMTP. Jedinou výjimkou z tohoto pravidla je případ, kdy se u adresy zjistí, že odkazuje na místního hostitele. V takovém případě bude předána řídicímu modulu.

  Program `smail` rozeznává IP-adresy, které jsou napsané jak s použitím tečkové notace, tak s využitím právoplatných názvů hostitelů v případě, že se dají rozložit pomocí systémového volání *gethostbyaddr(3)*. Například adresa **scrooge@[149.76.12.4]** je platnou adresou, i když je to značně neobvyklá poštovní adresa pro uživatele **scrooge** na hostiteli **quark.physics.groucho.edu**.

  Jestliže je váš počítač v Internetu, nepředstavují tyto směrovače nic z toho, co byste mohli potřebovat, protože nepodporují záznamy typu MX. Co máte v takovémto případě dělat, se dozvíte dále.

- Pokud existuje databáze cest `/usr/lib/smail/paths`, pokusí se program `smail` vyhledat v tomto souboru cílového hostitele (bez řetězce **.uucp**, který je uveden na konci). Pošta poslaná na adresu, která vyhovuje tomuto směrovači, bude doručena prostřednictvím protokolu UUCP s využitím cesty, jež byla nalezena v databázi.

- Adresa hostitele (bez řetězce **.uucp**, který je uveden na konci) bude porovnána s výstupem příkazu `uuname`, aby se zjistilo, zda je cílový hostitel skutečně sousedem UUCP. V tom případě bude zpráva doručena prostřednictvím transportního modulu pro protokol UUCP.

- Pokud adresa neodpovídala ani jednomu z výše uvedených směrovačů, bude doručena chytřejšímu hostiteli. Cesta na chytřejšího hostitele je společně s používaným transportem nastavena v souboru `config`.

Tato implicitní nastavení fungují správně u mnoha jednoduchých nastavení, avšak selhávají v případě, kdy jsou směrovací požadavky trošku komplikovanější. Při podobných potížích si budete muset nainstalovat svůj vlastní soubor `routers`, který potlačí implicitní nastavení. Vzorový soubor `routers`, ze kterého byste mohli vyjít, je uveden v příloze B. Některé distribuce operačního systému Linux také přicházejí se skupinou konfiguračních souborů, které jsou upraveny tak, aby se s těmito problémy vypořádaly.

Pravděpodobně nejhorší problémy vyvstávají v situacích, kdy je váš hostitel provozován ve dvojím prostředí, jak se spojeními na bázi protokolu UUCP, tak i se spojeními pomocí modemu na bázi protokolu IP. V tom případě budete mít v souboru `hosts` názvy hostitelů, se kterými budete komunikovat jen příležitostně pomocí protokolu SLIP, takže program `smail` se bude pokoušet doručovat veškerou poštu těmto hostitelům prostřednictvím protokolu SMTP. To ale nebudete potřebovat, protože i když je spojení pomocí protokolu SLIP aktivováno pravidelně, bude posílání pošty po protokolu SMTP mnohem pomalejší, než po síti na bázi protokolu UUCP. Při implicitním nastavení neexistuje žádný způsob, jak tento problém programu `smail` vyřešit.

Tomuto problému se můžete vyhnout, necháte-li program `smail` zkontrolovat soubor `paths` dříve, než se bude dotazovat resolveru. Pak můžete do souboru `paths` vložit všechny hostitele, u kterých chcete explicitně definovat doručování pošty po síti na bázi protokolu UUCP. Pokud nechcete posílat prostřednictvím protokolu SMTP *žádné* zprávy, můžete označit jako komentáře veškeré směrovače používající resolver.

Další problém spočívá v tom, že implicitní nastavení neposkytuje správné směrování pošty v síti Internet, protože směrovač založený na resolveru nevyhodnocuje záznamy typu MX. Chcete-li povolit úplnou podporu pro směrování pošty v Internetu, označte tento směrovač jako komentář a odstraňte značku komentáře z toho směrovače, jenž používá místo resolveru službu BIND. Nicméně některé distribuce operačního systému Linux obsahují binární soubory programu `smail`, do nichž není zakompilována podpora služby BIND. Pokud povolíte službu BIND, ale v souboru `paniclog` najdete zprávu „`router inet_hosts: driver bind not found`“, budete si muset sehnat zdrojový kód k programu `smail` a tento program znovu zkompilovat (viz výše uvedená stať 14.2).

A nakonec není obvykle příliš vhodné používat ovladač `uuname`. První problém spočívá v tom, že tento ovladač vygeneruje v případě, že nemáte nainstalován protokol UUCP, chybu konfigurace, protože nenajde žádný příkaz `uuname`. Druhý problém nastane, máte-li v souboru `Systems` protokolu UUCP uvedeno více systémů, s nimiž máte skutečné poštovní spo-

jení. Těmito systémy mohou být hostitelé, s nimiž si pouze vyměňujete news, nebo hostitelé, ze kterých si občas stahujete nějaké soubory pomocí anonymní služby UUCP, avšak na druhou stranu si s nimi nevyměňujete žádné soubory.

Abyste tento problém vyřešili, můžete nahradit program `uuname` skriptem příkazového interpretu, který bude provádět jednoduchou funkci `exit 0`. Avšak obecnější řešení spočívá v úpravě souboru `routers` a odstranění tohoto ovladače.

14.7.1 Databáze paths

Program `smail` očekává, že nalezne databázi cest v souboru `paths`, který se nachází v adresáři `/usr/lib/smail`. Tento soubor je volitelný, takže pokud nechcete provádět žádné směrování cest, stačí tento soubor odstranit.

Soubor `paths` musí být setříděným souborem ASCII, který obsahuje záznamy, jež mapují názvy cílových systémů na cesty protokolu UUCP zadané pomocí vykřičníkové notace. Soubor je nutno setřídit, protože program `smail` používá k vyhledání požadovaného systému binární hledání. V tomto souboru nejsou povoleny komentáře a název systému musí být oddělen od vlastní cesty znakem tabulátoru. Databáze cest jsou podrobněji probrány ve 13. kapitole.

Pokud jste tento soubor vygenerovali ručně, měli byste se ujistit, že jste do něj vložili všechny právoplatné názvy daného systému. Je-li například nějaký systém známý jak pod prostým názvem pro protokol UUCP, tak i pod plně kvalifikovaným doménovým jménem, musíte do tohoto souboru přidat položku pro každý z těchto názvů. Soubor může být setříděn například tak, že ho předáte příkazu `sort`*(1)*.

Je-li však váš systém pouze koncovým systémem, nebudete potřebovat vůbec žádný soubor `paths`: stačí pouze nastavit v souboru `config` vlastnosti týkající se chytřejšího hostitele a dále můžete nechat veškeré směrování pošty na svém poštovním doručovateli.

14.8 Doručování zpráv na místní adresy

Nejběžněji místní adresa odpovídá přihlašovacímu jménu uživatele. V takovémto případě je zpráva doručena do jeho poštovní schránky do adresáře `/var/spool/mail/`*user*, kde *user* je přihlašovací jméno daného uživatele. Dalšími případy je použití přezdívek, názvů poštovních konferencí nebo uživatelské směrování pošty. V těchto případech se místní adresa rozloží do nového seznamu adres, které mohou být buď místní, nebo vzdálené.

Kromě těchto „normálních“ adres se může program `smail` starat i o další typy místních destinací zpráv, jimiž mohou být například názvy souborů nebo převedení zpráv příkazům. Toto nejsou adresy v pravém slova smyslu, takže nemůžete poslat poštu například na „adresu“ **/etc/passwd@vbrew.com**; tyto adresy jsou korektní pouze v případě, že byly převzaty ze směrování nebo ze souborů přezdívek.

Za `název souboru` se považuje vše, co začíná znakem lomítka (/) nebo znakem vlnovky (~). Znak vlnovky odkazuje na domovský adresář příslušného uživatele a je přípustný pouze v případě, že byl název souboru převzat ze souboru `.forward` nebo ze směrovacího záznamu v poštovní schránce (viz níže). Pokud se doručuje do souboru, připojí program `smail` zprávu k danému souboru nebo v případě nutnosti tento soubor vytvoří.

Příkaz, na nějž má být daná zpráva převedena, může být jakýkoliv unixový příkaz, před kterým je uveden symbol propojení (|). Tento symbol zapříčiní, že program *smail* předá daný příkaz společně s jeho argumenty příkazového interpretu, ale tentokrát už bez uvozujícího symbolu (|). Vlastní zpráva je předána na standardní vstup tohoto příkazu.

Například k předávání poštovní konference do místní diskusní skupiny můžete použít skript příkazového interpretu s názvem `gateit` a potom nastavit místní přezdívku používající „`|gateit`“, která bude doručovat veškeré zprávy z poštovního seznamu do daného skriptu.

Obsahuje-li vyvolání příkazu bílé místo, musí být daný příkaz uzavřen do uvozovek. Vzhledem k objevujícím se bezpečnostním problémům jsou učiněna opatření, aby se příkaz nespustil v případě, že byla adresa získána nějakým pochybným způsobem (například pokud do souboru s přezdívkami, z něhož byla adresa získána, může zapisovat naprosto každý).

14.8.1 Místní uživatelé

Nejběžnějším případem místní adresy je uvedení poštovní schránky uživatele. Poštovní schránka je umístěna v adresáři `/var/spool/mail` a má přiřazeno jméno uživatele. Jejím vlastníkem je daný uživatel, který patří do skupiny **mail** a má nastavena přístupová práva 660. Pokud poštovní schránka neexistuje, pak ji program `smail` vytvoří.

Poznamenejte si, že ačkoliv je v současnosti adresář `/var/spool/mail` standardním místem pro umístění souborů poštovní schránky, mohou mít některé poštovní programy předkompilovány odlišné cesty, například do adresáře `/usr/spool/mail`. Pokud na vašem počítači neustále selhává doručování pošty, pak můžete zkusit, zda nepomůže vytvoření symbolického odkazu do adresáře `/var/spool/mail`.

Program `smail` vyžaduje existenci dvou adres: **MAILER-DAEMON** a **Postmaster**. U nedoručitelné pošty se vygeneruje skoková zpráva. Kopie této zprávy je poslána na účet **postmaster** k prozkoumání (to je kvůli možnému výskytu nějakého konfiguračního problému). Adresa **MAILER-DAEMON** se používá u skokové zprávy jako adresa odesílatele.

Pokud tyto adresy ve vašem systému neodpovídají korektním účtům, namapuje program `smail` účet **MAILER-DAEMON** na účet **postmaster**, resp. účet **postmaster** na účet **root**. Toto mapování můžete obvykle potlačit tak, že přezdívku k účtu **postmaster** přidělíte komukoliv, kdo je zodpovědný za údržbu poštovního softwaru.

14.8.2 Směrování

Uživatel si může přesměrovat svou poštu tak, že ji pošle na alternativní adresu. Program `smail` k tomuto účelu podporuje dvě metody. Jednou z možností je vložit následující položku na první řádek souboru poštovní schránky:

`Forward to` *recipient*,...

Tato položka zajistí, že bude veškerá příchozí pošta poslána všem příjemcům v zadaném seznamu. Další možností je vytvoření souboru `.forward` v domovském adresáři daného uživatele, který bude obsahovat čárkami oddělený seznam příjemců. U tohoto způsobu směrování jsou přečteny a zanalyzovány všechny uvedené řádky.

Všimněte si, že v souboru `.forward` můžete použít libovolný typ adresy. Praktická ukázka souboru `.forward`, který lze použít v případě dovolené, by mohla vypadat asi takto:

```
janet, "|vacation"
```

První uvedená adresa vždy doručí příchozí zprávu do poštovní schránky uživatele **janet**, avšak příkaz `vacation` pošle odesilateli krátké oznámení.

14.8.3 Soubory s přezdívkami

Program `smail` může spravovat takové soubory s přezdívkami, které jsou kompatibilní se soubory s přezdívkami, jež zná program `sendmail` vytvořený na univerzitě v Berkeley. Položky v souboru s přezdívkami mají následující syntaxi:

```
alias:   recipients
```

Pole `recipients` se skládá z čárkami odděleného seznamu adres, jež budou nahrazeny přezdívkou. Seznam příjemců může pokračovat i na více řádcích, pokud následující řádek začíná znakem tabulátor.

Dále existuje speciální vlastnost, která umožní programu *smail* přebírat poštovní konference ze souboru přezdívek: zadáte-li místo adresy příjemce řetězec „:include:filename“, přečte program `smail` soubor `filename` a použije seznam příjemců z tohoto souboru.

Hlavní soubor s přezdívkami má název `/usr/lib/aliases`. Umožníte-li zápis do tohoto souboru komukoliv, pak program `smail` nebude doručovat žádné zprávy příkazům příkazového interpretu, které jsou v tomto souboru uvedeny. Nyní následuje vzorový soubor `aliases`:

```
# soubor /usr/lib/aliases file pro vbrew.com
hostmaster: janet
postmaster: janet
usenet: phil
# Diskusní skupina development
development: joe, sue, mark, biff
          /var/mail/log/development
owner-development: joe
# Oznámení jsou zasílána všem zaměstnancům
announce: :include: /usr/lib/smail/staff,
          /var/mail/log/announce
owner-announce: root
# Brána mezi poštovní konferencí a lokální diskusní skupinou
ppp-list: "|/usr/local/lib/gateit local.lists.ppp"
```

Pokud se při doručování na adresu vygenerovanou ze souboru `alias` vyskytne nějaká chyba, pokusí se program `smail` poslat kopii chybové hlášky „vlastníkovi přezdívky“. Pokud například při doručování zprávy podle poštovního seznamu **development** neuspěje doručení zprávy uživateli **biff**, bude kopie chybové zprávy poslána odesilateli a stejně tak i na účty **postmaster** a **owner-development**. Jestliže adresa vlastníka neexistuje, nebude již vygenerována žádná další zpráva.

Při doručování do souborů nebo při vyvolávání programů uvedených v souboru `aliases` přejde z bezpečnostních důvodů program `smail` na účet uživatele **nobody**. Zvláště při doručování do souborů by mohly nastat skutečné bezpečnostní problémy. Například u výše uvedeného souboru musí být vlastníkem log-souborů uživatel **nobody**. Tento uživatel musí mít i právo do nich zapisovat. V opačném případě by doručování do těchto souborů neuspělo.

14.8.4 Poštovní konference

Poštovní konference mohou být spravovány nejen pomocí souboru *aliases*, ale také prostřednictvím souborů nacházejících se v adresáři `/usr/lib/smail/lists`. Poštovní konference s názvem *nag-bugs* je popsána v souboru `lists/nag-bugs`. Tento soubor by měl obsahovat čárkami oddělené adresy členů. Seznam může být uveden na několika řádcích a komentáře musí být uvozeny znakem (#).

Ke každé poštovní konferenci by měl existovat uživatel (nebo přezdívka) s názvem **owner-**`listname`, kde `listname` je název dané poštovní konference; veškeré chyby, které se vyskytnou při rozkládání adresy, budou oznámeny tomuto uživateli. Tato adresa se také používá jako adresa odesilatele u všech odcházejících zpráv. Je uvedena v hlavičce zprávy v poli `Sender:`.

14.9 Přenosy na bázi protokolu UUCP

V programu `smail` je zakompilováno množství transportů, které využívají balík UUCP. V prostředí protokolu UUCP jsou zprávy posílány na dalšího hostitele obvykle pomocí příkazu `rmail`. Zpráva je tomuto příkazu předána ze standardního vstupu a adresa na obálce mu je předána na příkazovém řádku. Na vašem hostiteli by měl být příkaz `rmail` symbolickým odkazem na příkaz `smail`.

Při předání zprávy transportu na bázi protokolu UUCP převede program `smail` cílovou adresu do vykřičníkové notace, jež se používá v prostředí protokolu UUCP. Například adresa **user@host** bude převedena na adresu **host!user**. Výskyt všech adresových operátorů '*%*' bude zachován, takže adresa **user%host@gateway** bude převedena na adresu **gateway!user%host**. Avšak samotný program `smail` nikdy takovouto adresu nevygeneruje.

Alternativně může program `smail` pomocí protokolu UUCP posílat a přijímat dávky protokolu BSMTP. U protokolu BSMTP se zabalí jedna nebo více zpráv do jediné dávky, jež bude obsahovat příkazy, které by spustil místní mailer v případě, že by se realizovalo skutečné spojení pomocí protokolu SMTP. Protokol BSMTP se často užívá v sítích typu store-and-forward (tj. například u sítí na bázi protokolu UUCP), aby se ušetřilo místo na disku. Vzorový soubor `transports` uvedený v dodatku B obsahuje přenos nazvaný `bsmtp`, který generuje v adresáři fronty dílčí dávky protokolu BSMTP. Později musí být dílčí dávky vloženy do konečných dávek pomocí skriptu příkazového interpretu, jenž do nich přidá příkazy *HELO* a *QUIT*.

Chcete-li u nějakého konkrétního spojení pomocí protokolu UUCP povolit transport *bsmtp*, musíte použít tzv. soubory metod (`method` files) (nahlédněte prosím na manuálovou stránku `smail`*(5)*, kde získáte více informací). Máte-li pouze jedno spojení na bázi protokolu UUCP

a pokud používáte směrovač chytřejšího hostitele, pak posílání dávek protokolu SMTP povolíte tak, že konfigurační proměnné *smart_transport* přiřadíte místo hodnoty *uux* hodnotu *bsmtp*.

Chcete-li přijímat dávky protokolu SMTP po spojení na bázi protokolu UUCP, musíte se ujistit, že máte příkaz na opětovné rozložení dávek, kterému bude vzdálený systém posílat své dávky. Pokud vzdálený systém také používá program `smail`, pak budete muset vytvořit odkaz `rsmtp` na program `smail`. Jestliže na vzdáleném počítači běží program `sendmail`, měli byste dodatečně nainstalovat skript příkazového interpretu s názvem `/usr/bin/bsmtp`, který bude provádět jednoduchý příkaz „`exec rsmtp`“ (protože symbolický odkaz nebude fungovat).

14.10 Přenosy na bázi protokolu SMTP

V současnosti podporuje program `smail` ovladač protokolu SMTP, který se používá k doručování pošty po sítích na bázi protokolu TCP.[6] Program `smail` je schopen doručit zprávu na libovolný počet adres nacházejících se na jednom samostatném hostiteli, jehož název je zadán buď jako plně kvalifikované doménové jméno, jež může být rozloženo pomocí síťového softwaru, nebo je zadán s respektováním tečkové notace, v tom případě ale musí být tento název uveden v hranatých závorkách. Adresy resolvované pomocí nějakého z ovladačů směrovačů BIND, *gethostbyname(3)* nebo *gethostbyaddr(3),* budou obvykle doručeny přenosu na bázi protokolu SMTP.

Ovladač protokolu SMTP se pokusí okamžitě spojit se vzdáleným hostitelem na portu *smtp*, který je uveden v souboru `/etc/services`. Není-li tento port dosažitelný nebo dojde-li k překročení časového limitu stanoveného pro dané spojení, bude nový pokus o doručení této zprávy učiněn někdy později.

Doručování v síti Internet vyžaduje, aby směrování na cílového hostitele bylo zadáno ve formě *route-addr*, která byla popsána ve 13. kapitole a nikoliv pomocí vykřičníkové notace.[7] Proto program `smail` převede adresu **user%host@gateway**, kde brána **gateway** je dosažitelná pomocí adresy **host1!host2!host3**, na adresu zdrojového směrování **<@host2,@host3:-user%host@gateway>**, jež bude poslána jako adresa na obálce na adresu **host1**. Chcete-li tyto transformace adres povolit (společně s ovladačem služby BIND), musíte v souboru `transports` upravit položku týkající se ovladače *smtp*. Vzorový soubor `transports` je uveden v dodatku B.

6 Autoři označují tuto podporu jako „jednoduchou“. Předesílají, že další verze programu `smail` bude úplně přepracovaná, což výrazně zlepší správu protokolu SMTP.

7 Nicméně v síti Internet se snažíme zabránit používání směrování. Místo toho by měla být použita plně kvalifikovaná doménová jména.

14.11 Omezení názvů hostitelů

Někdy je žádoucí odchytit veškeré nekvalifikované názvy hostitelů (to jsou takové názvy, které neobsahují název domény), které jsou zadány jako adresy odesilatele nebo příjemce. Je to užitečné například při provozování brány mezi dvěma sítěmi, z nichž jedna vyžaduje plně kvalifikovaná doménová jména. Na bráně spojující Internet se sítí na bázi protokolu UUCP by měly být nekvalifikované názvy hostitelů implicitně namapovány do domény **uucp**. Jakékoliv jiné úpravy adres jsou problematické.

Soubor `/usr/lib/smail/qualify` sděluje programu `smail`, které názvy domén se mají přiřadit jednotlivým názvům hostitelů. Položky souboru `qualify` jsou složeny z názvu hostitele, který začíná ve sloupci jedna a za ním následuje název domény. Řádky obsahující znak (#), který je uveden jako první znak, který není bílým místem, jsou považovány za komentáře. Položky se prohledávají v tom pořadí, v jakém jsou zadány.

Pokud neexistuje soubor `qualify`, nebude prováděno žádné překládání nekvalifikovaných názvů.

Speciální název hostitele * zastupuje všechny hostitele, což vám umožní mapovat do implicitní domény všechny hostitele, o kterých jste se v souboru `qualify` nezmínili. Tento název by měl být poslední položkou souboru.

Ve společnosti Virtual Brewery bylo pro všechny hostitele nastaveno, aby používali v adrese odesilatele plně kvalifikované názvy domény. Nekvalifikované adresy příjemce se překládají do implicitní domény **uucp**, takže v souboru *qualify* je nutná pouze jediná položka.

```
# /usr/lib/smail/qualify, poslední změna 12.února, 1994 janet
#
*              uucp
```

Kapitola

15

Program sendmail+IDA

15.1 Úvod do programu sendmail+IDA

Říká se, že *skutečným* správcem Unixu se stanete až v okamžiku, když se vám povede upravit soubor `sendmail.cf`. Ale také se říká, že musíte být blázen, když to chcete zkoušet podruhé.:-)

Program `sendmail` je neuvěřitelně výkonný nástroj. Většina lidí ho chápe jen velmi obtížně a ještě hůře se ho učí. Jakýkoliv program, jehož kompletní manuál (manuál k programu `sendmail` vydalo nakladatelství O'Reilly and Associates) má 792 stran, zcela pochopitelně vyděsí většinu lidí.

Program `sendmail+IDA` je odlišný. Odstraňuje nutnost úprav záhadného souboru `sendmail.cf` a správcům umožňuje definovat směrování a konfiguraci adres, které jsou specifické pro daný systém pomocí poměrně jednoduše srozumitelných podpůrných souborů tzv. *tabulek*. Zvolíte-li program `sendmail+IDA`, může vám to ušetřit spoustu hodin práce a stresu.

V porovnání s ostatními hlavními poštovními transportními agenty neexistuje asi nic, co by se nedalo pomocí programu `sendmail+IDA` udělat rychleji a jednodušeji. S jeho pomocí provedete poměrně jednoduše věci, které je potřeba vykonat před spuštěním normálního internetového systému nebo normálního systému na bázi protokolu UUCP. Konfigurace, jež jsou za normálních okolností velice obtížné, se v něm dají jednoduše vytvořit a spravovat.

V době sestavování této publikace je prostřednictvím anonymní služby FTP na adrese **vixen.cso.uiuc.edu** dostupná verze *`sendmail5.67b+IDA1.5`*. Je sestavena tak, aby na platformě operačního systému Linux nevyžadovala žádnou aktualizaci.

Všechny konfigurační soubory potřebné pro kompilaci zdrojového kódu programu `sendmail+IDA`, instalaci a spuštění pod operačním systémem Linux jsou obsaženy v balíku `newspak-2.2.tar.gz`. Ten je dostupný prostřednictvím anonymní služby FTP na adrese **sunsite.unc.edu** v adresáři `/pub/Linux/system/Mail`.

15.2 Konfigurační soubory – přehled

Tradiční program sendmail se nastavuje pomocí systémového konfiguračního souboru (obvykle je to soubor `/etc/sendmail.cf` nebo `/usr/lib/sendmail.cf`), který se nepodobá žádnému z jazyků, s nimiž jste se mohli až doposud setkat. Editace souboru *`sendmail.cf`* tak, aby se program sendmail choval požadovaným způsobem, může být hořkou zkušeností.

S programem `sendmail+IDA` je toto nelidské úsilí již věcí minulosti, protože všechny konfigurační volby jsou ovládány prostřednictvím tabulek, jež mají poměrně snadno pochopitelnou syntaxi. Tyto volby se konfigurují prostřednictvím programu `m4` (procesor maker) nebo programu `dbm` (databázový procesor), jenž se spustí na spoustě datových souborů prostřednictvím souboru `Makefile`, který je dodán společně se zdrojovými kódy.

V souboru `sendmail.cf` definujete pouze implicitní chování systému. Ve skutečnosti se všechny speciální úpravy uskutečňují prostřednictvím spousty doplňkových tabulek. Je to lepší, než přímo editovat soubor `sendmail.cf`. Seznam všech tabulek programu `sendmail` je uveden na obrázku 15.1.

`mailertable`	Definuje speciální chování pro vzdálené hostitele nebo domény.
`uucpxtable`	Přikazuje, že pošta bude doručována pomocí protokolu UUCP pouze na hostitele, kteří mají adresu uvedenou ve formátu systému DNS.
`pathtable`	Definuje cesty na bázi protokolu UUCP na vzdálené hostitele nebo domény, které splňují vykřičníkovou notaci.
`uucprelays`	Omezí cesty v souboru cest pouze na známé vzdálené hostitele.
`genericfrom`	Převede vnitřní adresy na všeobecně použitelné adresy, které jsou pro ostatní svět viditelné.
`xaliases`	Převede všeobecně použitelné adresy na korektní vnitřní adresy anebo převede korektní vnitřní adresy na všeobecně použitelné adresy.
`decnhetxtable`	Převede adresy odpovídající standardu definovanému v RFC-822 na adresy používané v sítích DEC.

Obrázek 15.1
Podpůrné soubory programu `sendmail`

15.3 Soubor sendmail.cf

U programu `sendmail+IDA` se soubor `sendmail.cf` neupravuje přímo, ale generuje se z konfiguračního souboru programu *m4*, který poskytuje správce místního systému. V následujícím výkladu ho budeme označovat jako soubor `sendmail.m4`.

Tento soubor obsahuje jen několik definic, ale převážně odkazy na tabulky, kde se provádí skutečná konfigurační práce. Obvykle je nutné zadat pouze:

- Názvy cest a názvy souborů používaných v místním systému.
- Název nebo názvy, pod kterými je systém znám v oblasti elektronické pošty.
- Požadovaný implicitní mailer (možná i požadovaný chytřejší hostitel).

Existuje velké množství parametrů, které lze definovat za účelem dosažení požadovaného chování místního systému nebo k potlačení předkompilovaných konfiguračních voleb. Tyto konfigurační volby jsou uloženy v souboru `ida/cf/OPTIONS`, který se nachází v adresáři zdrojových programů.

Soubor `sendmail.m4` může být při minimální konfiguraci (pro protokol UUCP nebo protokol SMTP platí, že veškerá nemístní pošta je přenášena na přímo připojeného chytřejšího hostitele) dlouhý jen 10 až 15 řádků, nepočítáme-li komentáře.

15.3.1 Příklad souboru sendmail.m4

Dále je uveden soubor `sendmail.m4` pro hostitele **vstout** ve společnosti Virtual Brewery. Hostitel **vstout** používá protokol SMTP ke komunikaci se všemi hostiteli sítě LAN pivovaru a veškerou poštu určenou pro ostatní místa posílá pomocí protokolu UUCP na hostitele **moria**, což je brána k síti Internet.

15.3.2 Obvykle používané parametry v souboru esndmail.m4

V souboru `sendmail.m4` se vždy vyžaduje pouze několik parametrů; \pokud vám vyhovují implicitní volby, můžete ostatní položky vynechat. Následující stati popisují podrobněji každou položku, která je uvedena v ukázkovém souboru `sendmail.m4`.

Položky definující cesty

```
dnl #define(LIBDIR,/usr/local/lib/mail)dnl # Kde jsou ostatní soubory?
```

Parametr *LIBDIR* definuje adresář, kde program `sendmail+IDA` očekává konfigurační soubory, různé tabulky dbm a speciální místní definice. V typické binární distribuci je tento parametr vestavěn do binárního souboru programu `sendmail` a tudíž nemusí být v souboru `sendmail.m4` explicitně určen.

Výše uvedený příklad obsahuje řádky uvozené řetězcem *dnl*. Tyto řádky jsou vlastně považovány za komentáře a jsou zde uvedeny pouze z informativních důvodů.

Chcete-li změnit umístění podpůrných souborů, odstraňte z výše uvedeného řádku uvozující řetězec *dnl*, nastavte cestu do požadovaného adresáře a znovu vytvořte a nainstalujte soubor `sendmail.cf`.


```
dnl #------------------SAMPLE SENDMAIL.M4 FILE------------------
dnl # (the string 'dnl' is the m4 equivalent of commenting out a line)
dnl # you generally don't want to override LIBDIR from the compiled in
paths
dnl #define(LIBDIR,/usr/local/lib/mail)dnl  # where all support files go
define(LOCAL_MAILER_DEF, mailers.linux)dnl  # mailer for local delivery
define(POSTMASTERBOUNCE)dnl                 # postmaster gets bounces
define(PSEUDODOMAINS, BITNET UUCP)dnl       # don't try DNS on these
dnl #-------------------------------------------------------------
dnl #
define(PSEUDONYMS, vstout.vbrew.com vstout.UUCP vbrew.com)
dnl                                         # names we're known by
define(DEFAULT_HOST, vstout.vbrew.com)dnl   # our primary 'name' for mail
define(UUCPNAME,vstout)dnl                  # our uucp name
dnl #
dnl #-------------------------------------------------------------
dnl #
define(UUCPNODES, |uuname|sort|uniq)dnl     # our uucp neighbors
define(BANGIMPLIESUUCP)dnl                  # make certain that uucp
define(BANGONLYUUCP)dnl                     #  mail is trated correctly
define(RELAY_HOST, moria)dnl        # our smart relay host
define(RELAY_MAILER, UUCP-A)dnl             # we reach moria via uucp
dnl #
dnl #-------------------------------------------------------------
```

```
dnl #
dnl # the various dbm lookup tables
dnl #
define(ALIASES, LIBDIR/aliases)dnl          # system aliases
define(DOMAINTABLE, LIBDIR/domaintable)dnl  # domainize hosts
define(PATHTABLE, LIBDIR/pathtable)dnl      # paths database
define(GENERICFROM, LIBDIR/generics)dnl     # generic from addresses
define(MAILERTABLE, LIBDIR/mailertable)dnl  # mailers per host or domain
define(UUCPXTABLE, LIBDIR/uucpxtable)dnl    # paths to hosts we feed
define(UUCPRELAYS, LIBDIR/uucpreslays)dnl   # short-circuit paths
dnl #
dnl #-------------------------------------------------------------
dnl #
dnl # include the 'real' code that makes it all work
dnl # (provided with the source code)
dnl #
include(Sendmail.mc)dnl                     # REQUIRED ENTRY !!!
dnl #
dnl #------------------ END OF SAMPLE SENDMAIL.M4 FILE------------------
```

Obrázek 15.2

Vzorový soubor `sendmail.m4` pro hostitele **vstout**

Definice místního maileru

```
# mailer pro místní doručování
define(LOCAL_MAILER_DEF, mailers.linux)dnl
```

Většina operačních systémů disponuje speciálním programem, který se stará o místní doručování pošty. V binárním kódu programu sendmail je již vestavěno několik typických programů pro většinu hlavních variant operačního systému Unix.

V Linuxu se musí explicitně definovat patřičný místní mailer, protože místní doručovací program nemusí bezpodmínečně být součástí distribuce, kterou jste nainstalovali. To lze provést přidáním parametru *LOCAL_MAILER_DEF* v souboru `sendmail .m4`.

Například u běžně používaného programu `deliver`,[1] který poskytuje tuto službu, byste měli parametru *LOCAL_MAILER_DEF* přiřadit hodnotu `mailers.linux`.

[1] Program `deliver` napsal Chip Salzenberg (**chip%tct@ateng.com**). Tento program je součástí některých distribucí a navíc ho lze najít v běžných anonymních FTP archívech, například na adrese **ftp.uu.net**.

Dále by měl být do adresáře, na který ukazuje parametr *LIBDIR*, nainstalován soubor s názvem `mailers.linux`. Tento soubor explicitně definuje program `deliver` ve vnitřním maileru *Mlocal* s patřičnými parametry, které zajistí, že program `sendmail` bude korektně doručovat poštu adresovanou do místního systému. Nejste-li expertem v problematice programu sendmail, nebudete zřejmě chtít následující příklad nijak upravovat.

```
# -- /usr/local/lib/mail/mailers.linux -
#      místní mailery pro Linux
Mlocal, P=/usr/bin/deliver, F=SlsmFDMP, S=10, R=25/10, A=deliver $u
Mprog,  P=/bin/sh,          F=lsDFMeuP,   S=10, R=10, A=sh -c $u
```

V souboru `Sendmail.mc` existuje také vestavěná implicitní volba pro program *`deliver`*, kterou lze začlenit do konfiguračního souboru `sendmail.cf`. Chcete-li ji uvést, nesmíte použít soubor `mailers.linux`, ale místo něho musíte v souboru `sendmail.m4` definovat následující záznam:

```
dnl --- (soubor sendmail.m4) ---
define(LOCAL_MAILER_DEF, DELIVER)dnl  # mailer pro místní doručování
```

Naštěstí makro `Sendmail.mc` předpokládá, že je doručovací program nainstalován v adresáři `/bin`, což ale neplatí pro balík Slackware 1.1.1 (který ho nainstaluje do adresáře `/usr/bin`). V takovémto případě se budete muset s tímto problémem vypořádat buď tak, že vytvoříte symbolický odkaz, nebo tak, že znovu sestavíte ze zdrojového kódu doručovací program, který bude moci být uložen v adresáři `/bin`.

Vypořádání se s odmítnutou poštou

```
# odmítnutá pošta se pošle postmasterovi
define(POSTMASTERBOUNCE)dnl
```

Většina systémů zjistila, že je důležité, aby byla pošta poslána a doručena s téměř 100% úspěšností. I když prověřování log-souborů démona `syslogd`*(8)* je užitečné, potřebují obvykle správci místní pošty vidět hlavičky odmítnuté pošty, aby mohli zjistit, zda byla pošta nedoručitelná z důvodu chyby na straně uživatele, nebo z důvodu konfigurační chyby na jednom ze zúčastněných systémů.

Pokud definujete parametr *POSTMASTERBOUNCE*, bude kopie každé odmítnuté zprávy poslána osobě, která má v daném systému na starosti účet **Postmaster**.

Bohužel nastavení tohoto parametru bude mít vliv i na *text* zprávy, protože i ten je poslán na účet **Postmaster**. Tento text může obsahovat osobní věci týkající se lidí, kteří používají poštu ve vašem systému.

Poštmistři by proto neměli číst poštu (přesněji by měli používat skripty příkazového interpretu, které z odmítnutých příchozích zpráv vymažou text), která jim není určena.

Položky vztahující se k systému DNS

```
# pro tyto domény nepoužívej DNS
define(PSEUDODOMAINS, BITNET UUCP)dnl
```

Existuje několik známých sítí, na které jsou v poštovních adresách často odkazy, ale které nejsou korektní z hlediska systému DNS. Pokud definujete parametr *PSEUDODOMAINS*, zabráníte tak zbytečným pokusům při vyhledávání pomocí systému DNS, jež by stejně nikdy neuspěly.

Definice názvů, pod nimiž je místní systém známý

```
define(PSEUDONYMS, vstout.vbrew.com  vstout.UUCP vbrew.com
dnl                                          # naše jména
# naše primární poštovní jméno
define(DEFAULT_HOST, vstout.vbrew.com)dnl
```

Systémy chtějí často skrýt svoji pravou totožnost, chtějí sloužit jako poštovní brány nebo dostávat a zpracovávat poštu adresovanou na ‘staré‘ názvy, pod nimiž byly dříve známé.

Parametr *PSEUDONYMS* uvádí seznam všech názvů hostitelů, pro něž bude místní systém přijímat poštu.

Parametr *DEFAULT_HOST* uvádí název hostitele, který se zobrazí ve zprávě, jež byla vytvořena na místním hostiteli. Je důležité, aby byl tento parametr nastaven na korektní hodnotu, protože jinak by veškerá zpáteční pošta byla nedoručitelná.

Položky vztahující se k protokolu UUCP

```
define(UUCPNAME, vstout)dnl              # naše UUCP-jméno
define(UUCPNODES, |uuname|sort|uniq)dnl  # naši UUCP-sousedé
define(BANGIMPLIESUUCP)dnl               # je nutné pro korektní
define(BANGONLYUUCP)dnl                  # zpracování UUCP-pošty
```

Systém DNS často zná určité místo jiným názvem, než pod jakým ho zná síť na bázi protokolu UUCP. Parametr *UUCPNAME* vám dovoluje definovat odlišný název hostitele, který se objeví v hlavičkách odcházející pošty určené pro protokol UUCP.

Parametr *UUCPNODES* definuje příkazy, které vrátí seznam názvů hostitelů v systémech, se kterými jste přímo spojeni prostřednictvím spojení na bázi protokolu UUCP.

Parametry *BANGIMPLIESUUCP* a *BANGONLYUUCP* zajišťují, že pošta adresovaná pomocí syntaxe s vykřičníkovou notací, která je charakteristická pro protokol UUCP, bude zpracována po způsobu protokolu UUCP, a nikoliv podle běžnějšího chování systému DNS, který se nyní používá v Internetu.

Přenosové systémy a mailery

```
define(RELAY_HOST, moria)dnl      # náš chytřejší hostitel
define(RELAY_MAILER, UUCP-A)dnl   # cesta k hostiteli moria přes UUCP
```

Mnoho správců systému se nechce obtěžovat s takovou konfigurací, která by umožnila spojení se všemi sítěmi (a tedy i se všemi systémy) ve všech sítích po celém světě. Místo toho budou raději posílat veškerou výstupní poštu na další systém, o kterém ví, že je věru „chytřejší“.

Parametr *RELAY_HOST* definuje název hostitele tohoto chytřejšího sousedního systému. Název je definován pro protokol UUCP.

Parametr *RELAY_MAILER* definuje mailer, který se použije pro přenos zpráv na chytřejšího hostitele.

Je třeba poznamenat, že nastavení těchto parametrů způsobí, že vaše odcházející pošta bude směrována na tento vzdálený systém, což samozřejmě bude mít vliv na zatížení tohoto vzdáleného systému. Dříve, než nakonfigurujete váš systém tak, aby využíval vzdálený systém jako všeobecně použitelnou bránu, musíte získat od poštmistra tohoto systému explicitní souhlas.

Různé konfigurační tabulky

```
define(ALIASES, LIBDIR/aliases)dnl            # systémové přezdívky
define(DOMAINTABLE, LIBDIR/domaintable)dnl    # tabulka domaintable
define(PATHTABLE, LIBDIR/pathtable)dnl        # databáze cest
define(GENERICFROM, LIBDIR/generics)dnl
define(MAILERTABLE, LIBDIR/mailertable)dnl    # mailery pro hostitele
  a domény
define(UUCPXTABLE, LIBDIR/uucpxtable)dnl
define(UUCPRELAYS, LIBDIR/uucprelays)dnl
```

Pomocí těchto maker můžete změnit adresáře, ve kterých program `sendmail+IDA` hledá různé tabulky dbm, jež definují „skutečné“ chování systému. Obvykle je moudré nechat tyto tabulky v adresáři *LIBDIR*.

Hlavní soubor Sendmail.mc

```
include(Sendmail.mc)dnl                    # NUTNÝ ZÁZNAM!!!
```

Autoři programu `sendmail+IDA` dodávají soubor `Sendmail.mc`, který obsahuje pravou „podstatu" toho, co se později stane se souborem `sendmail.cf`. Pravidelně jsou uvolňovány nové verze, které obsahují opravené chyby a přidávají další funkce, aniž by vyžadovaly plnou verzi a překompilování programu `sendmail` ze zdrojových kódů.

Je velice důležité, abyste tento soubor *neupravovali*.

Jaké položky jsou tedy vyžadovány?

Pokud nepoužíváte žádnou z doplňkových tabulek dbm, doručí program `sendmail+IDA` poštu v souladu s nastavenou volbou *DEFAULT_MAILER* (eventuálně v souladu s nastavenými volbami *RELAY_HOST* a *RELAY_MAILER*), která je definována v souboru *sendmail.m4*. Tento soubor se používá ke generování souboru `sendmail.cf`. Toto chování lze jednoduše potlačit pomocí záznamů uvedených v tabulce `domaintable` nebo `uucpxtable`.

Všeobecný systém umístěný v Internetu a využívající služeb systému DNS nebo systém založený pouze na protokolu UUCP, jenž směruje veškerou poštu pomocí protokolu UUCP na chytřejšího hostitele, který je definován prostřednictvím volby *RELAY_HOST*, nebude pravděpodobně vyžadovat žádné specifické záznamy v tabulkách.

Všechny systémy by měly mít ve skutečnosti nastaveny makra *DEFAULT_HOST* a *PSEUDONYMS*, která definují kanonický název daného systému a jeho přezdívky, a dále by měly mít nastaveno makro *DEFAULT_MAILER*. Máte-li nastaveny volby *RELAY_HOST* a *RELAY_MAILER*, nebudete muset tyto implicitní volby nastavovat, protože budou fungovat automaticky.

U hostitelů na bázi protokolu UUCP bude pravděpodobně potřeba nastavit parametr *UUCPNAME* na jejich oficiální název pro protokol UUCP. Dále pravděpodobně nastaví parametry *RELAY_HOST* a *RELAY_MAILER*, které povolují směrování přes chytřejšího hostitele prostřednictvím poštovní brány. Používaný poštovní transport je definován pomocí volby *RELAY_MAILER* a obvykle by měl být pro sítě na bázi protokolu UUCP nastaven na hodnotu *UUCP-A*.

Pokud váš systém pracuje pouze na bázi protokolu SMTP a komunikuje prostřednictvím systému DNS, měli byste změnit parametr *DEFAULT_MAILER* na hodnotu *TCP-A* a dále smazat řádky s parametry *RELAY_MAILER* a *RELAY_HOST*.

15.4 Přehled tabulek programu sendmail+IDA

Program `sendmail+IDA` podporuje množství tabulek, které vám umožňují potlačit implicitní chování programu `sendmail` (definované v souboru `sendmail.m4`) a dále vám umožňují definovat speciální chování pro zvláštní situace, vzdálené systémy a sítě. Tyto tabulky jsou později zpracovány pomocí programu `dbm` s využitím souboru `Makefile`, který je součástí distribuce.

Většina systémů bude potřebovat pouze několik těchto tabulek. Možná, že nebudete potřebovat vůbec žádné tabulky. Pokud váš systém tyto tabulky nevyžaduje, bude asi nejjednodušší, když je vytvoříte jako soubory s nulovou velikostí (pomocí příkazu `touch`) a použijete implicitní soubor `Makefile`, který najdete v adresáři *LIBDIR*. Bude to lepší, než kdybyste museli editovat vlastní soubor `Makefile`.

15.4.1 Tabulka mailertable

Tabulka `mailertable` definuje speciální zacházení s konkrétními hostiteli nebo doménami, které vychází z názvu vzdáleného hostitele nebo z názvu sítě. U systémů v síti Internet se často používá k výběru pomocného hostitele nebo k přenosu pošty nebo brány, jejichž prostřednictvím se můžete spojit se vzdálenými sítěmi. Dále se používá k definici konkrétního protokolu (UUCP nebo SMTP), který se bude používat. Systémy na bázi protokolu UUCP obvykle nebudou muset tento soubor používat.

Pořadí položek v souboru je důležité. Program `sendmail` čte tento soubor směrem shora dolů a zpracovává zprávu podle prvního odpovídajícího pravidla, které najde. Takže je dobré umístit nejjednoznačnější pravidla na začátek souboru a obecnější pravidla na konec souboru.

Předpokládejme, že chcete směrovat veškerou poštu určenou pro oddělení počítačových věd na Groucho Marx University pomocí protokolu UUCP na bránu **ada**. Aby to bylo možné, musíte mít v tabulce `mailertable` položku, která vypadá asi následovně:

```
# (soubor mailertable)
#
# poštu pro doménu.cs.groucho.edu posílej přes UUCP-hostitele ada
UUCP-A,ada          .cs.groucho.edu
```

Dále předpokládejme, že chcete z důvodu rozlišení adres a z důvodu doručování veškerou poštu určenou pro rozsáhlejší doménu **groucho.edu** směrovat na zvláštní bránu **bighub**. Doplněné záznamy by mohly v tabulce `mailertable` vypadat asi takto:

```
# (soubor mailertable)
#
```

```
# poštu pro doménu cs.groucho.edu posílej přes UUCP-hostitele ada
UUCP-A,ada.cs.groucho.edu
#
# poštu pro doménu cs.groucho.edu posílej přes UUCP-hostitele bighub
UUCP-A,bighub.cs.groucho.edu
```

Jak jsme se již zmínili, je pořadí velmi důležité. Kdybychom zaměnili pořadí těchto dvou pravidel, pak by se veškerá pošta určená pro doménu **.cs.groucho.edu** směrovala na obecnější bránu **bighub** místo toho, aby putovala na jednoznačně určenou bránu **ada**, což bylo naším záměrem.

```
# (soubor mailertable)
#
# poštu pro doménu.groucho.edu posílej přes UUCP-hostitele bighub
UUCP-A,bighub .groucho.edu
#
# (další řádek je zbytečný,
# protože je použito předchozí pravidlo)
UUCP-A,ada    .cs.groucho.edu
```

Ve výše uvedených příkladech je definován mailer *UUCP-A*, který sděluje programu `sendmail`, aby používal k doručování pošty protokol UUCP s hlavičkami respektujícími doménový systém.

Čárka mezi mailerem a vzdáleným systémem sděluje maileru, že má zprávy směrovat na bránu **ada**, která se postará o rozlišení adres a doručení zpráv.

Záznamy v tabulce `mailertable` mají následující syntaxi:

```
mailer delimiter relayhost                    host_or_domain
```

Existuje několik druhů mailerů. Rozdíl mezi nimi obvykle spočívá v tom, jakým způsobem zacházejí s adresami. Běžné mailery jsou *TCP-A* (pro protokol TCP/IP s adresami typickými pro síť Internet), *TCP-U* (pro protokol TCP/IP s adresami typickými pro sítě na bázi protokolu UUCP) a *UUCP-A* (pro sítě na bázi protokolu UUCP, ale s adresami typickými pro síť Internet).

Znak nacházející se na levé straně řádky tabulky `mailertable`, který odděluje mailer od části s názvem hostitele, definuje způsob, jakým bude tabulka `mailertable` upravovat příslušnou adresu. Jedinou věcí, kterou je zde třeba upravit, je přepsat adresu na obálce (aby se pošta mohla dostat do vzdáleného systému). Přepisování čehokoliv jiného se nedoporučuje, protože by to mohlo vést k poškození konfigurace pošty.

! Znak vykřičník odstraní název hostitele příjemce předtím, než je zpráva směrována na mailer. Tento oddělovač by se měl použít v tom případě, kdy chcete nařídit odeslání pošty do špatně nakonfigurovaného vzdáleného systému.

, Znak čárka adresu vůbec neupravuje. Zpráva je většinou směrována pomocí zadaného maileru na zadanou bránu.

: Znak dvojtečka odstraní název hostitele příjemce pouze v případě, že mezi vaším hostitelem a cílovým hostitelem existují nějací mezilehlí hostitelé. Bude-li tedy adresa **foo!bar!joe**, bude hostitel **foo** odstraněn, avšak bude-li adresa **xyzzy!janet**, pak zůstane nezměněná.

15.4.2 Tabulka uucpxtable

Pošta na hostitele s plně kvalifikovanými doménovými jmény je obvykle doručena buď prostřednictvím směrování charakteristického pro síť Internet (protokol SMTP) s využitím systému DNS, nebo prostřednictvím přenosových hostitelů (bran). Tabulka `uucpxtable` zajistí doručení pomocí směrování na bázi protokolu UUCP, protože převede doménový název na bezdoménový název vzdáleného hostitele, který je charakteristický pro protokol UUCP.

Tato tabulka se často používá v případech, kdy jste dopravcem pošty pro daný systém nebo doménu, nebo když chcete poslat poštu pomocí přímého a spolehlivého spojení na bázi protokolu UUCP a nikoliv přes implicitní mailer a několik mezilehlých systémů a sítí, kde by eventuálně mohlo dojít k několika skokům.

Systémy na bázi protokolu UUCP komunikující se svými UUCP-sousedy, kteří používají doménové poštovní hlavičky, by měly používat tento soubor, aby si vynutili doručení pošty prostřednictvím přímého spojení typu point-to-point na bázi protokolu UUCP, protože jinak by musely používat méně přímé směrování prostřednictvím parametrů *RELAY_MAILER* a *RELAY_HOST* nebo pomocí parametru *DEFAULT_MAILER*.

Internetové systémy, které nepoužívají komunikaci na bázi protokolu UUCP, nebudou muset tabulku `uucpxtable` používat.

Předpokládejme, že poskytujete doručovací službu systému s názvem **sesame.com** prostřednictvím systému DNS a systému **sesame** prostřednictvím map protokolu UUCP. Do tabulky `uucpxtable` budete muset vložit následující položku, abyste poštu směrovanou na jejich hostitele mohli poslat po přímém spojení na bázi protokolu UUCP.

```
#=============== /usr/local/lib/mail/uucpxtable ==============
# Pošta poslaná na adresu joe@sesame.com bude přepsána na sesame!joe
# a poté doručena přes UUCP
```

```
#
sesame sesame.com
#
#-----------------------------------------------------------
```

15.4.3 Tabulka pathtable

Tabulka `pathtable` slouží k definici explicitního směrování na vzdálené hostitele nebo sítě. Soubor tabulky `pathtable` by měl mít syntaxi souboru cest `paths` a měl by být setříděn podle abecedy. Dvě pole na každé řádce musí být oddělena znakem tabulátoru, jinak by si mohl program `dbm` stěžovat.

Většina systémů nebude potřebovat žádné záznamy v tabulce `pathtable`.

```
#=============== /usr/local/lib/mail/pathtable ==============
#
# this is a pathalias-style paths file to let you kick mail to
# UUCP neighbors to the direct UUCP path so you don't have to
# go the long way through your smart host that takes other traffic
#
# you want real tabs on each line or m4 might complain
#
# route mail through one or more intermediate sites to a remote
# system using UUCP-style addressing
#
sesame!ernie!%s                 ernie
#
# forwarding to a system that is a UUCP neighbor of a reachable
# internet site.
```

```
#

swim!%s@gcc.groucho.edu         swim

# The following sends all mail for two networks through different

# gateways (see the leading '.' ?).

# In this example, "uugate" and "byte" are specific systems that
server

# as mail gateways to the .UUCP and .BITNET pseudo-domains respec-
tively

#

%s@uugate.groucho.edu           .UUCP
byte!%s@mail.shift.com          .BITNET
```


15.4.4 Tabulka domaintable

Tabulka `domaintable` se používá k dosažení jistého chování při vyhledávání pomocí systému DNS. Správci povoluje vytvořit zkrácené názvy, které lze použít u běžných odkazů na systémy nebo domény. Tyto zkrácené názvy tabulka `domaintable` automaticky nahradí odpovídajícími plnými názvy. Dále může být tabulka použita k nahrazení nekorektních názvů hostitelů nebo domén „korektními“ informacemi.

Většina systémů nebude potřebovat žádné záznamy v tabulce `domaintable`.

Následující příklad ukazuje, jak nahradit nekorektní adresu, kterou se snaží lidé v poště používat, korektní adresou:

```
#=============== /usr/local/lib/mail/domaintable ==============
#
#
brokenhost.correct.domain                brokenhost.wrong.domain
#
#
#=============== konec domaintable ============================
```

15.4.5 Tabulka aliases

Přezdívky umožňují hned několik věcí:

- Pro směrování pošty poskytují zkrácený název nebo veřejný název, takže pošta může být doručena jedné nebo více osobám.
- Vyvolávají program, kterému je poštovní zpráva předána na vstup.
- Posílají poštu do souboru.

Všechny systémy vyžadují přezdívky pro účty **Postmaster** a **MAILER-DAEMON**, aby splňovaly standardy definované v RFC.

Při definování přezdívek, které vyvolávají programy nebo zapisují do programů, buďte vždy opatrní, protože program `sendmail` má zpravidla nastavenou hodnotu uid na **root**.

Změny provedené v souboru `aliases` se projeví až po spuštění následujícího příkazu, který vytvoří požadované tabulky dbm:

```
# /usr/lib/sendmail -bi
```

Totéž lze také obvykle provést spuštěním příkazu `newaliases` z programu `cron`.

Detaily týkající se poštovních přezdívek můžete nalézt na manuálové stránce *aliases(5)*.

```
#=============== /usr/local/lib/mail/aliases ==============
#
# příklad obvyklých typů přezdívek
#
usenet:         janet                 # přezdívka pro jednu osobu
admin:          joe, janet            # přezdívka pro více lidí
newspak-users:  :include:/usr/lib/lists/newspak
                                      # příjemci jsou v souboru
changefeed:     | /usr/local/lib/gup  # přezdívka, která spustí
                                        program
complaints:     /var/log/complaints   # přezdívka, která zapíše
                                        poštu do souboru
#
# Následující přezdívky jsou podle RFC povinné.
# Je důležité, aby se pošta směrovaná na tyto adresy dostala do
  přihrádky někoho, kdo čte poštu pravidelně.
```

```
#
postmaster:      root                    # nutné
MIALER-DAEMON:   postmaster              # nutné
#
#-------------------------------------------------------------
```

15.4.6 Zřídka používané tabulky

Následující tabulky jsou sice dostupné, ale používají se jen velmi zřídka. Chcete-li o nich získat více detailů, nahlédněte prosím do dokumentace, která je součástí zdrojového kódu programu `sendmail+IDA`.

`uucprelays` — Soubor `uucprelays` se používá ve zvláště známých systémech ke „zkrácení" cesty protokolu UUCP. Je to lepší, než používat víceskokové nebo nespolehlivé cesty, které jsou vygenerovány z map protokolu UUCP tabulkou `pathalias`.

`genericfrom` a `xaliases` — Soubor `genericfrom` skrývá před okolním světem názvy místních uživatelů a jejich adresy. Toho může dosáhnout tak, že převede jména místních uživatelů na obecné adresy odesilatele, které neodpovídají interním jménům uživatelů.

Přidružená utilita s názvem `xalparse` automatizuje generování tabulky `genericfrom` a souboru `aliases`, takže převod jak odcházejícího, tak i přicházejícího jména uživatele se může dít z hlavního souboru `xaliases`.

`decnetxtable` — Tabulka `decnetxtable` přepisuje doménové adresy na adresy vyhovující standardu, který se používá v sítích DEC. Tento převod je podobný přepisu bezdoménových adres na doménové adresy, které lze použít v sítích s protokolem SMTP.

15.5 Instalace programu sendmail

V této stati se zaměříme na to, jak lze nainstalovat typickou distribuci binárního kódu programu `sendmail+IDA` a projdeme si kroky, které je zapotřebí učinit, aby byl tento program lokalizovaný a funkční.

Současnou verzi binárního kódu programu `sendmail+IDA` pro platformu Linux můžete získat na adrese **sunsite.unc.edu** v adresáři `/pub/Linux/system/mail`. Dokonce i když máte dřívější verzi programu `sendmail`, vřele doporučuji přejít na verzi `sendmail5.67b+IDA1.5`[2], protože všechny požadované aktualizace specifické pro platformu Linux jsou obsaženy ve zdrojovém kódu „vanilla sources" a navíc bylo odstraněno i několik závažných bezpečnostních děr, které se nacházely ve verzích, jež byly uvolněny před 1. prosincem 1993.

Jestliže sestavujete program `sendmail` ze zdrojového kódu, měli byste postupovat podle instrukcí uvedených v souborech `README`, které jsou součástí distribuce zdrojového kódu. Aktuální verze zdrojového kódu programu `sendmail+IDA` jsou k dispozici na adrese **vixen.cso.uiuc.edu**. Chcete-li sestavit program `sendmail+IDA` na linuxové platformě, budete k tomu potřebovat i konfigurační soubory specifické pro operační systém Linux, které se nacházejí v balíku `newspak-2.2.tar.gz`. Tento balík je k dispozici na adrese **sunsite.unc.edu** v adresáři `/pub/Linux/system/Mail`.

Pokud jste již dříve nainstalovali program `smail` nebo nějakého jiného poštovního doručovacího agenta, budete pravděpodobně chtít z bezpečnostních důvodů odstranit (nebo přejmenovat) veškeré soubory vztahující se k programu `smail`.

15.5.1 Rozbalení distribuce binárního kódu

Nejdříve musíte rozbalit archívní soubor na nějaké bezpečné místo:

```
$ gunzip -c sendmail5.65b+IDA1.5+mailx5.3b.tgz | tar xvf -
```

Máte-li „novější" verzi programu `tar`, například z poslední verze balíku Slackware, stačí pravděpodobně napsat pouze `tar -zxvf filename.tgz`, kde `filename` je název souboru, a dostanete stejný výsledek.

Při rozbalování daného archívu se vytvoří adresář s názvem `sendmail5.65b+IDA1.5-+mailx5.3b`. V tomto adresáři naleznete kompletní instalaci programu `sendmail+IDA` a navíc v něm bude i binární kód programu `mailx`, což je uživatelský agent. Všechny cesty k souborům nacházející se pod tímto adresářem určují umístění, kam by se měly soubory nainstalovat, takže je vhodné navrhnout takový příkaz `tar`, který přesune tyto soubory na požadované místo.

```
# cd sendmail5.65b+IDA1.5+mailx5.3b
# tar cf - . | (cd /; tar xvvpoof -)
```

2 Pozn. korektora: Poslední verze programu sendmail má číslo 8.9.1.

15.5.2 Sestavení souboru sendmail.cf

Abyste mohli sestavit soubor `sendmail.cf`, který bude přizpůsoben potřebám vašeho systému, musíte nejdříve napsat soubor `sendmail.m4`, jenž zpracujete pomocí programu `m4`. V adresáři `/usr/local/lib/mail/CF` naleznete ukázkový soubor s názvem `sample.m4`. Zkopírujte ho do souboru *název_vašeho_hostitele*`.m4` a upravte ho tak, aby odrážel situaci panující ve vašem systému.

Vzorový soubor je nastaven pro systém založený pouze na protokolu UUCP, který má doménové hlavičky a komunikuje s chytřejším hostitelem. U systémů podobných tomuto nastavení se musí v tomto souboru provést jen mírné úpravy.

V této stati vám poskytnu pouze krátký přehled maker, které musíte pozměnit. Chcete-li kompletní popis jejich funkce, nahlédněte prosím do dřívějších statí, kde jsme se zabývali souborem `sendmail.m4`.

LOCAL_MAILER_DEF Určuje soubor, ve kterém jsou definovány mailery pro místní doručování pošty. Chcete-li vědět, co vše do tohoto souboru patří, nahlédněte prosím do stati „Definice místního maileru“.

PSEUDONYMS Definuje všechny názvy, pod kterými je známý váš místní hostitel.

DEFAULT_HOST Sem zadejte svoje plně kvalifikované doménové jméno. Tento název se objeví jako název vašeho hostitele ve veškeré odcházející poště.

UUCPNAME Sem zadejte svůj nekvalifikovaný název hostitele.

RELAY_HOST a *RELAY_MAILER*
Jestliže komunikujete se svým chytřejším hostitelem pomocí protokolu UUCP, nastavte parametr *RELAY_HOST* na název pro protokol UUCP, který bude definovat vašeho „chytřejšího přenosového“ souseda v síti UUCP. Jestliže chcete doménové hlavičky, nastavte mailer na hodnotu *UUCP-A*.

DEFAULT_MAILER Jestliže jste v Internetu a komunikujete pomocí systému DNS, měli byste tento parametr nastavit na hodnotu *TCP-A*. Řetězec *TCP-A* sdělí programu `sendmail`, aby použil mailer *TCP-A*, který doručuje poštu pomocí protokolu SMTP, a aby použil pro adresu na obálce normální adresu kompatibilní se standardy uvedenými v RFC. Systémy v Internetu pravděpodobně nebudou muset definovat parametry *RELAY_HOST* a *RELAY_MAILER*.

Chcete-li vytvořit soubor `sendmail.cf`, spusťte následující příkaz:

```
# make název_vašeho_hostitele.cf
```

Tento příkaz zpracuje váš soubor *název_vašeho_hostitele*`.m4` a vytvoří z něho soubor *název_vašeho_hostitele*`.cf`.

Dále musíte otestovat, zda vytvořený konfigurační soubor provádí přesně to, co jste od něj očekávali. Testování konfiguračního souboru bude vysvětleno v následujících dvou statích.

Jste-li teď již spokojeni s chováním konfiguračního souboru, zkopírujte ho pomocí následujícího příkazu na patřičné místo:

```
# cp název_vašeho_hostitele.cf /etc/sendmail.cf
```

Od tohoto okamžiku může váš program `sendmail` začít pracovat. Do příslušného startovního souboru (obvykle je to soubor `/etc/rc.inet2`) vložte následující řádku. Chcete-li tento proces rozběhnout okamžitě, můžete ručně spustit následující příkaz.

```
# /usr/lib/sendmail -bd -q1h
```

15.5.3 Testování konfiguračního souboru sendmail.cf

Chcete-li spustit program `sendmail` v „testovacím“ režimu, spusťte ho s parametrem `-bt`. Implicitní konfigurační soubor se nazývá `sendmail.cf`, ten je také nainstalován ve vašem systému. Alternativní soubor můžete otestovat pomocí volby `-C` *filename*, kde *filename* je název daného souboru.

V následujících příkladech budeme testovat soubor `vstout.cf`, což je konfigurační soubor vygenerovaný ze souboru `vstout.m4`, který je zobrazen na obrázku 15.2.

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
>
```

Následující testy zaručí, že je program `sendmail` schopen doručit veškerou poštu uživatelům ve vašem systému. Ve všech případech by měl být výsledek testu stejný a měl by ukazovat na místní název systému s nastaveným mailerem *LOCAL*.

Nejprve otestujme, jakým způsobem by měla být doručena pošta místnímu uživateli.

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 me
rewrite: ruleset  3   input: me
rewrite: ruleset  7   input: me
rewrite: ruleset  9   input: me
rewrite: ruleset  9  returns: < me >
rewrite: ruleset  7  returns: < >, me
rewrite: ruleset  3  returns: < >, me
rewrite: ruleset  0   input: < >, me
rewrite: ruleset  8   input: < >, me
rewrite: ruleset 20   input: < >, me
rewrite: ruleset 20  returns: < >, @ vstout . vbrew . com , me
rewrite: ruleset  8  returns: < >, @ vstout . vbrew . com , me
rewrite: ruleset 26   input: < >, @ vstout . vbrew . com , me
rewrite: ruleset 26  returns: $# LOCAL @$ vstout . vbrew . com $: me
rewrite: ruleset  0  returns: $# LOCAL @$ vstout . vbrew . com $: me
>
```

Výstup ukazuje, jak program `sendmail` vnitřně zpracovává adresu. Tato adresa je předána různým skupinám pravidel, které ji analyzují, spustí postupně další skupiny pravidel a rozloží ji na jednotlivé komponenty.

V našem příkladu jsme předali adresu **me** skupinám pravidel 3 a 0 (to je význam hodnot 3, 0, které jsou uvedeny před adresou). Poslední řádka ukazuje rozloženou adresu, která je výsledkem použití skupiny pravidel 0. Tato rozložená adresa se skládá z maileru, pomocí něhož by měla být daná zpráva doručena, a ze jmen hostitele a uživatele, které jsou maileru předány.

Dále otestujme poštu adresovanou uživateli vašeho systému, adresace bude splňovat syntaxi protokolu UUCP.

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 vstout!me
rewrite: ruleset  3  input: vstout ! me
```

```
[...]
rewrite: ruleset   0   returns: $# LOCAL $@ vstout . vbrew . com $: me
>
```

Dále otestujme adresaci pošty uživateli vašeho systému, adresace bude splňovat syntaxi typickou pro Internet, tedy pošta bude adresována na váš plně kvalifikovaný název hostitele.

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 me@vstout.vbrew.com
rewrite: ruleset  3 input: me @ vstout . vbrew . com
[...]
rewrite: ruleset  0 returnes: $# LOCAL $@ vstout . vbrew . com $: me
>
```

Výše uvedené dva testy byste měli zopakovat pro každý název, který jste uvedli v parametrech *PSEUDONYMS* a *DEFAULT_NAME*, jež se nacházejí v souboru `sendmail.m4`.

A konečně otestujme, zda můžete posílat poštu na svého přenosového hostitele (na svou bránu).

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 fred@moria.com
rewrite: ruleset   3 input:   fred @ moria . com
rewrite: ruleset   7 input:   fred @ moria . com
rewrite: ruleset   9 input:   fred @ moria . com
rewrite: ruleset   9 returns: < fred > @ moria . com
rewrite: ruleset   7 returns: < @ moria . com > , fred
rewrite: ruleset   3 returns: < @ moria . com > , fred
rewrite: ruleset   0 input:   < @ moria . com > , fred
rewrite: ruleset   8 input:   < @ moria . com > , fred
rewrite: ruleset   8 returns: < @ moria . com > , fred
rewrite: ruleset  29 input:   < @ moria . com > , fred
rewrite: ruleset  29 returns: < @ moria . com > , fred
rewrite: ruleset  26 input:   < @ moria . com > , fred
```

```
rewrite: ruleset  25  input:    < @ moria . com > , fred
rewrite: ruleset  25  returns:  < @ moria . com > , fred
rewrite: ruleset   4  input:    < @ moria . com > , fred
rewrite: ruleset   4  returns:  fred @ moria . com
rewrite: ruleset  26  returns:  < @ moria . com > , fred
rewrite: ruleset   0  returns:  @# UUCP-A $@ moria $:
  < @ moria . com > , fred
>
```

15.5.4 Všechno dohromady – integrované testování souboru sendmail.cf a jeho tabulek

V tomto okamžiku máte již ověřeno, že pošta dodržuje požadované implicitní chování a že můžete posílat i přijímat korektně adresovanou poštu. K dokončení instalace vám už chybí jen vytvořit patřičné tabulky dbm, abyste získali požadované konečné výsledky.

Po vytvoření tabulky nebo tabulek, které jsou potřebné pro váš systém, je musíte zpracovat pomocí programu `dbm`. To lze provést tak, že v adresáři s tabulkami napíšete příkaz `make`.

Pokud provozujete systém pouze na bázi protokolu UUCP, nemusíte vytvářet žádnou z tabulek, o kterých se zmiňuje soubor `README.linux`. Stačí pouze vytvořit soubory tabulek s nulovou velikostí, aby soubor `Makefile` pracoval správně.

Pokud provozujete systém pouze na bázi protokolu UUCP a pokud komunikujete i s jinými systémy, než je váš chytřejší hostitel, budete muset pro každý z těchto systémů vložit do tabulky `uucpxtable` (protože jinak by pošta adresovaná na ně šla přes chytřejšího hostitele) příslušnou položku a znovu spustit příkaz `dbm`, aby se tabulka `uucpxtable` zaktualizovala.

Nejprve se musíte ubezpečit, že pošta poslaná na tyto systémy přes přenosového hostitele, který je definován pomocí parametru *RELAY_HOST*, bude poslaná pomocí maileru, který je definován v parametru *RELAY_MAILER*.

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 fred@sesame.com
rewrite: ruleset    3      input:   fred @ sesame . com
rewrite: ruleset    7      input:   fred @ sesame . com
rewrite: ruleset    9      input:   fred @ sesame . com
```

```
rewrite: ruleset    9    returns:   < fred > @ sesame . com
rewrite: ruleset    7    returns:   < @ sesame . com > , fred
rewrite: ruleset    3    returns:   < @ sesame . com > , fred
rewrite: ruleset    0      input:   < @ sesame . com > , fred
rewrite: ruleset    8      input:   < @ sesame . com > , fred
rewrite: ruleset    8    returns:   < @ sesame . com > , fred
rewrite: ruleset    29     input:   < @ sesame . com > , fred
rewrite: ruleset    29   returns:   < @ sesame . com > , fred
rewrite: ruleset    26     input:   < @ sesame . com > , fred
rewrite: ruleset    25     input:   < @ sesame . com > , fred
rewrite: ruleset    25   returns:   < @ sesame . com > , fred
rewrite: ruleset    4      input:   < @ sesame . com > , fred
rewrite: ruleset    4    returns:   fred @ sesame . com
rewrite: ruleset    26   returns:   < @ sesame . com > , fred
rewrite: ruleset    0    returns:   @# UUCP-A $@ sesame $: < @ sesa-
me . com > , fred
>
```

Máte-li i jiné sousedy v síti UUCP, než je váš přenosový hostitel definovaný pomocí parametru *RELAY_HOST*, musíte se ujistit, že se pošta adresovaná na tyto hostitele bude chovat korektně. Pošta adresovaná pomocí vykřičníkové notace na hostitele, se kterým komunikujete na bázi protokolu UUCP, by měla být poslána přímo na tohoto hostitele (pokud to výslovně nezakážete prostřednictvím položky uvedené v tabulce `domaintable`). Předpokládejme, že hostitel **swim** je vaším přímým sousedem sítě UUCP. V tom případě, pokud programu `sendmail` pošlete adresu **swim!fred**, by se měl objevit následující výsledek:

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 swim!fred
rewrite: ruleset  3   input: swim ! fred
[...vynecháno...]
rewrite: ruleset  0   returnes: $# UUCP $@ swim $: < >, fred
>
```

Pokud máte v tabulce `uucpxtable` uvedeny položky, jež přikazují doručit poštu pomocí protokolu UUCP určitým sousedům sítě UUCP, kteří používají u své pošty doménové hlavičky splňující standardy definované v síti Internet, je potřeba otestovat také tyto transportní cesty.

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 dude@swim.2birds.com
rewrite: ruleset 3 input: dude @ swim . 2birds . com
[...vynecháno...]
rewrite: ruleset 0 returns: $# UUCP $@ swim . 2 birds $: < >, dude
>
```

15.6 Administrace a jednoduché poštovní triky

Když jsme nyní probrali teorii konfigurace, instalace a testování programu `sendmail+IDA`, zastavme se na chvíli u věcí, které se často stávají v každodenním životě správce pošty.

Vzdálené systémy se někdy porouchají. Modemy nebo telefonní linky selžou, definice systému DNS jsou díky chybě člověka nastaveny chybně. Sítě naprosto neočekávaně spadnou. V takových situacích musí správci pošty vědět, jak mají rychle, efektivně a *bezpečně* zasáhnout, aby pošta stále mohla chodit přes alternativní směrování, dokud vzdálené systémy nebo poskytovatelé služeb neobnoví normální provoz.

Zbytek této kapitoly budeme věnovat řešení nejčastěji se vyskytujících případů „nouzových stavů elektronické pošty“.

15.6.1 Směrování pošty na přenosového hostitele

Směrujete-li poštu pro konkrétního hostitele nebo doménu na určitého přenosového hostitele, používáte k tomu obvykle tabulku `mailertable`.

Chcete-li například směrovat poštu pro doménu **backwood.org** na její bránu UUCP s názvem **backdoor**, pak vložíte do tabulky `mailertable` následující záznam:

```
UUCP-A,backdoor    backwood.org
```

15.6.2 Vynucené poslání pošty do špatně zkonfigurovaných vzdálených systémů

Hostitelé v Internetu mají často problémy s posláním pošty do špatně zkonfigurovaných vzdálených systémů. Existuje několik variant tohoto problému, avšak obecným příznakem těchto problémů je, že pošta je vzdáleným systémem odmítnuta nebo se do něj vůbec nedostane.

Tyto problémy staví správce místního systému do těžké pozice, protože vaše uživatele zpravidla nezajímá, že osobně nespravujete všechny systémy na celém světě (nebo že nevíte, jak přimět vzdáleného správce k tomu, aby tento problém odstranil). Oni pouze vědí, že se jejich pošta nemůže prokousat na druhou stranu k příslušnému příjemci a že vy jste vhodnou osobou, které by si mohli stěžovat.

Konfigurace vzdáleného systému je jejich problém, a ne váš. Ve všech případech se ale ujistěte, že *nepoškodíte* svůj systém jen kvůli tomu, abyste mohli komunikovat se špatně nakonfigurovaným vzdáleným systémem. Nemůžete-li sdělit poštmistrovi vzdáleného systému, aby v časově přijatelném období opravil svoji konfiguraci, máte jen dvě možnosti.

- Většinou je možné si úspěšně vynutit poslání pošty do vzdáleného systému, ale protože je vzdálený systém špatně nakonfigurovaný, nemusí na vzdáleném konci správně fungovat odpovědi…ale to už je vlastně problém správce vzdáleného systému.

  Špatné hlavičky na obálce vaší odcházející pošty můžete pro daného hostitele nebo doménu upravit pouze prostřednictvím položky v tabulce `domaintable`. Ta způsobí opravu nekorektních informací v poště pocházející z vašeho systému:

  ```
  braindead.correct.domain.com        braindead.wrong.domain.com
  ```

- Často špatně zkonfigurované systémy odmítnou přijetí pošty zpět do systému odesilatele a zobrazí působivou zprávu „that mail isn't for this site" (tato pošta není určena pro daný systém), protože tyto systémy nemají ve své konfiguraci správně nastaveny své přezdívky nebo jejich ekvivalenty pomocí parametru *PSEUDONYMS*. U veškeré pošty adresované na takovéto systémy je možné odstranit z adresy na obálce všechny názvy hostitelů a informace o doménách.

  Znak vykřičník (*!)* uvedený v následujícím záznamu tabulky `mailertable` doručí poštu na patřičný vzdálený systém. Tato pošta se bude jevit jejich programu `sendmail` stejně tak, jako kdyby byla vytvořena na daném vzdáleném systému. Pamatujte si, že tato položka mění pouze adresu na obálce, takže ve zprávě se bude i nadále zobrazovat správná zpáteční adresa.

  ```
  TCP!braindead.correct.domain.com    braindead.wrong.domain.com
  ```

Bez ohledu na to, zda se vám povede poslat poštu do špatně nakonfigurovaného systému, nebudete mít žádnou záruku, že uživatelé ze vzdáleného systému budou moci na vaši zprávu odpovědět (pamatujte, že adresy jsou poškozené...). Avšak v takovém případě si budou spíše stěžovat uživatelé vzdáleného systému svým správcům, nežli vaši uživatelé vám.

15.6.3 Jak dosáhnou toho, aby byla pošta poslána prostřednictvím protokolu UUCP

V ideálním světě (z perspektivy sítě Internet) by měli všichni hostitelé uvedeny záznamy v doménové jmenné službě (v systému DNS) a posílali by poštu s plně kvalifikovanými názvy hostitelů.

Pokud budete muset komunikovat s takovým systémem na bázi protokolu UUCP, můžete zajistit posílání pošty po spojení na bázi protokolu UUCP typu point-to-point a nikoliv prostřednictvím implicitního maileru, který by v podstatě pomocí tabulky `uucpxtable` vytvořil z plně kvalifikovaného názvu hostitele nekvalifikovaný název hostitele.

Chcete-li si vynutit doručení na bázi protokolu UUCP na adresu **sesame.com**, měli byste do tabulky `uucpxtable` vložit následující záznam:

```
# uprav adresu sesame.com a pro doručení použij UUCP
sesame     sesame.com
```

Výsledkem je, že program `sendmail` zjistí (na základě parametru *UUCPNODES*, který se nachází v souboru `sendmail.m4`), že jste přímo spojeni se vzdáleným systémem a proto bude řadit poštu tak, aby se mohla doručit pomocí protokolu UUCP.

15.6.4 Zákaz doručování pošty na bázi protokolu UUCP

Občas se dostanete i do opačné situace. Systémy mohou mít často spoustu přímých spojení na bázi protokolu UUCP, které se používají jen zřídka, nebo které nejsou natolik spolehlivé a trvale dostupné jako implicitní mailer nebo přenosový hostitel.

Například v okolí města Seattle existuje spousta systémů, které si vyměňují různé distribuce operačního systému Linux prostřednictvím anonymní služby UUCP. Tato výměna se ale děje pouze v době, kdy je uvolněna nová verze distribuce. Tyto systémy komunikují pomocí protokolu UUCP pouze tehdy, když je to zapotřebí, takže je zpravidla rychlejší a spolehlivější posílat poštu přes několik spolehlivějších skoků prostřednictvím veřejných (vždy dostupných) přenosových hostitelů.

Doručování pošty na bázi protokolu UUCP na hostitele, s nímž jste přímo propojeni, zabráníte velmi snadno. Pokud má vzdálený systém plně kvalifikovaný název domény, můžete do tabulky `domaintable` přidat obdobnou položku:

```
# neposílej poštu svým sousedům přes UUCP
snorkel.com        snorkel
```

Tento záznam nahradí každý výskyt názvu pro protokol UUCP plně kvalifikovaným doménovým jménem (FQDN) a tím zabrání splnění řádky s parametrem *UUCPNODES*, která je definována v souboru `sendmail.m4`. Výsledkem bude obvykle to, že se pošta pošle na základě hodnot parametrů *RELAY_HOST* a *RELAY_MAILER* (nebo na základě hodnoty parametru *DEFAULT_MAILER*).

15.6.5 Spuštění programu sendmail v režimu okamžitého zpracování fronty

Chcete-li zprávy čekající ve frontě zpracovat okamžitě, zadejte pouze příkaz `/usr/lib/runq`. Tento příkaz vyvolá program `sendmail` s patřičně nastavenými volbami, které zajistí, že program `sendmail` spustí frontu nevyřízených úkolů okamžitě a nebude čekat na další plánované spuštění.

15.6.6 Zápis poštovních statistik

Mnoho správců systému (a lidí, kteří pro ně pracují) se zajímá o množství pošty poslané do místního systému, z místního systému nebo přes místní systém. Existuje několik způsobů, jak sledovat kvantitu poštovní dopravy.

- Součástí programu `sendmail` je utilita s názvem `mailstats`, která čte soubor `/usr/local/lib/mail/sendmail.st` a vypisuje počet zpráv a množství bajtů, které přenesl každý z mailerů definovaných v souboru `sendmail.cf`. Aby se mohly zapisovat statistiky o odeslané poště, musí být tento soubor ručně vytvořen správcem místního systému. Pokud odstraníte a znovu vytvoříte soubor `sendmail.st`, pak se součty jednotlivých statistik vymažou. Jeden z možných způsobů je následující:

  ```
  # cp /dev/null /usr/lib/local/mail/sendmail.st
  ```

- Pravděpodobně nejlepší způsob, který zpracovává kvalitní statistiky bez ohledu na to, kdo používá poštu a jaké množství pošty se posílá do místního systému, z místního systému nebo přes místní systém, představuje povolení poštovních ladicích informací. To lze provést prostřednictvím démona `syslogd`*(8)*. Zpravidla je nutné spustit soubor démona `/etc/syslogd` ze svého systémového startovacího souboru (což byste stejně měli udělat) a přidat řádek do souboru `/etc/syslog.conf`*(5)*, který vypadá asi následovně:

  ```
  mail.debug                                    /var/log/syslog.mail
  ```

  Používáte-li soubor `mail.debug` a máte-li poměrně velký objem pošty, může soubor log nabýt značné velikosti. Výstupní soubory z démona `syslogd` se zpravidla musí v pravidelných intervalech nahrazovat nebo „promazávat“ prostřednictvím démona `crond`*(8)*.

Existuje množství běžně dostupných utilit, které vytvářejí souhrn z výstupu poštovních log-souborů vytvořených démonem `syslogd`. Jedněmi z nejznámějších utilit je perlový skript `syslog-stat.pl`, který je součástí distribuce zdrojového kódu programu `sendmail+IDA`.

15.7 Kombinace a srovnávání jednotlivých distribucí binárního kódu

Konfigurace přenosu elektronické pošty a doručovacích agentů nemá žádný přesný standard a navíc ani nemá „jednotnou strukturu adresářů“.

Z toho důvodu je nutné se ujistit, že všechny odlišné části systému (USENET news, pošta, protokol TCP/IP) souhlasí s umístěním doručovacího programu pro místní poštu (například programy `lmail`, `deliver` atd.), doručovacího programu pro vzdálenou poštu (například program `rmail`) a poštovního transportního programu (například program `sendmail` nebo `smail`). Tyto předpoklady nejsou zpravidla uvedeny, i když použití příkazu `strings` vám může pomoci zjistit, jaké soubory a adresáře se očekávají. Následuje několik problémů, se kterými jsme se v minulosti setkali u některých běžně dostupných distribucí binárního kódu a zdrojového kódu operačního systému Linux.

- Některé verze distribuce balíku NET-2 protokolu TCP/IP mají definované služby pro program `umail` a nikoliv pro program `sendmail`.
- Existuje několik importovaných verzí programů *elm* a *mailx*, které hledají doručovacího agenta pod názvem `/usr/bin/smail` a nikoliv pod názvem `sendmail`.
- V programu `sendmail+IDA` je vestavěn místní mailer pro program `deliver`, avšak program `sendmail+IDA` očekává, že se tento program bude nacházet v adresáři `/bin` a nikoliv v adresáři `/usr/bin`, což je adresář typický pro operační systém Linux.

Abyste neměli při kompilaci všech poštovních klientů ze zdrojových kódů potíže, doporučujeme raději místo tohoto postupu vytvořit mezi jednotlivými programy symbolické odkazy.

15.8 Kde získat více informací

Existuje mnoho míst, kde byste mohli hledat informace o programu `sendmail`. Máte-li zájem o jejich seznam, nahlédněte prosím do dokumentu Linux Mail HOWTO, který je pravidelně posílán na adresu **comp.answers**. Tento informační bulletin je také dostupný prostřednictvím anonymní služby FTP na adrese **rtfm.mit.edu**. Avšak nejlepším zdrojem informací

jsou zdrojové kódy k programu `sendmail+IDA`. Podívejte se do souborů `DBM-GUIDE`, `OPTIONS` a `Sendmail.mc`, které se nachází v podadresáři `ida/cf` adresáře se zdrojovými kódy.

Kapitola

16

Síťové news (Netnews)

16.1 Historie Usenetu

Myšlenka síťových news se zrodila v roce 1979, kdy dva absolventi univerzity, Tom Truscott a Jim Ellis, začali uvažovat o využití protokolu UUCP ke spojení počítačů za účelem výměny informací mezi uživateli Unixu. Ve státě North Carolina vytvořili malou síť tvořenou třemi počítači.

Původně byl provoz sítě obsluhován několika skripty příkazového interpretu (které byly později přepsány do jazyka C), ale ty se nikdy nedostaly na veřejnost. Rychle je nahradily tzv. „A“ news, které byly prvním veřejným vydáním softwaru news.

Systém „A“ news nebyl navržen pro denní práci s více články ve skupině. Když začal jejich objem narůstat, rozhodli se ho Mark Horton a Matt Glickman přepsat, a výsledek svého snažení nazvali vydání „B“ (tzv. Bnews). První veřejné vydání Bnews mělo číslo verze 2.1 a stalo se tak v roce 1982. Postupně byl systém rozšiřován o některé nové rysy. Současná verze Bnews má číslo 2.11. Pomalu však zastarává, nehledě k tomu, že oficiální správce přešel na INN.

O další přepis se postarali Geoff Colyer a Hanry Spencer v roce 1987; šlo o verzi „C“, neboli C News. Následovala spousta oprav, z nichž nejvýznamnější bylo vydání C News Performance Release. V systémech, které spravují velké množství diskusních skupin, se režie způsobená častým spouštěním programu `relaynews`, který je zodpovědný za odesílání příchozích článků jiným hostitelům, stává významnou. Verze Performance Release přidává novou volbu `relaynews`, která umožňuje spouštět stejnojmenný program, v režimu démona, kdy tento program umístí sám sebe na pozadí.

Verze Performance Release je verzí C News, která je v současné době součástí většiny linuxových balíků.

Všechna vydání až po verzi „C“ jsou určena především pro sítě UUCP, i když je lze stejně dobře používat i v jiných prostředích. Efektivní přenos v sítích typu TCP/IP, DECNet nebo jim podobných vyžaduje nové schéma. Z toho důvodu byl v roce 1986 zaveden protokol NNTP (Network News Transfer Protocol – Protokol pro přenos síťových news). Je založen na síťových spojeních a specifikuje množství příkazů pro interaktivní předávání a získávání článků.

Na Síti existuje spousta aplikací založených na protokolu NNTP. Jednou z nich je i balík `nntpd`, jehož autory jsou Brian Barber a Phil Lapsley. Kromě jiného může sloužit také jako služba zajišťující čtení news hostitelům v rámci lokální počítačové sítě. Balík `nntpd` byl navržen jako doplněk balíků Bnews nebo C News, které měl rozšířit o vlastnosti protokolu NNTP.

Jiným balíkem vycházejícím z protokolu NNTP je `INN`, neboli Internet News. Nejedná se o systém front end, ale spíše o plnohodnotný systém news. Obsahuje důmyslného démona pro přenos news, který je schopen efektivně spravovat několik souběžných spojení NNTP a z toho důvodu je vhodným serverem pro mnoho systémů připojených k Internetu.

16.2 Čím je Usenet v každém případě?

Jedním z ohromujících faktů ohledně Usenetu je skutečnost, že není součástí žádné organizace ani jej neřídí žádná centralizovaná autorita pro správu sítě. Ve skutečnosti je součástí usenetové lidové tradice, kterou bez ohledu na technický popis nelze nijak definovat. Můžete jen říci, čím není. Kdybyste měli po ruce knihu Brendana Kehoe „Zen and the Art of the Internet“ (je dostupná on-line nebo přes Prentice Hall, viz [Kehoe92]), našli byste zde zábavný seznam usenetových „nevlastností“.

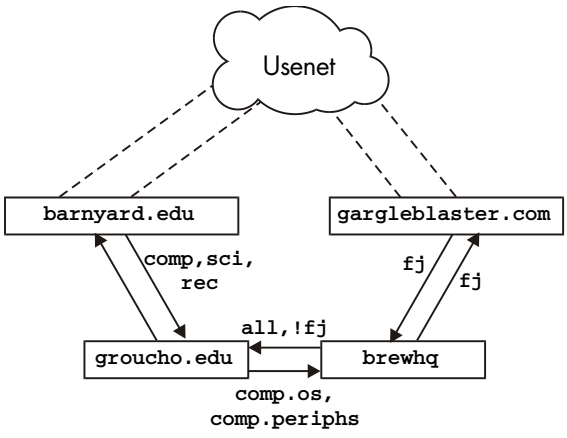
Budeme-li riskovat, že to bude znít hloupě, mohli bychom Usenet definovat jako sdružení vzájemně oddělených míst, které si mezi sebou vyměňují news. Aby se váš systém stal součástí Usenetu, potřebujete nalézt jiný takový systém a sjednat si dohodu s jeho vlastníky nebo správci o výměně news. Poskytování news jinému systému se někdy také říká „krmení“, z čehož vychází další obecný axiom usenetové filosofie: „Dej mu nažrat a jsi v něm.“

Nejmenší jednotkou usenetových news je článek. Jedná se o zprávu, kterou uživatel „odešle do sítě“. Aby s ním mohl systém news pracovat, je opatřen administrativní informací, takzvanou hlavičkou článku. Ta je podobná formátu poštovní hlavičky specifikované standardem Internet mail standard RFC 822 v tom, že je tvořena několika řádky textu, z nichž každý začíná názvem pole ukončeným dvojtečkou, za nímž následuje hodnota tohoto pole.[1]

[1] Formát zpráv usenetových news specifikuje standard RFC 1036, „Standard for interchange of USENET messages“.

Články jsou postupovány jedné nebo více diskusním skupinám (newsgroup). Diskusní skupina se zabývá články týkajícími se jednoho tématu. Všechny diskusní skupiny jsou hierarchicky organizovány, přičemž název každé skupiny zároveň udává její pozici v této hierarchii. Například z názvu diskusní skupiny **comp.os.linux.announce** může každý poznat, že slouží pro oznámení týkající se počítačového operačního systému jménem Linux.

Tyto články si pak mezi sebou vymění všechna usenetová místa, která mají zájem o news z příslušné diskusní skupiny. Pokud dvě místa souhlasí s výměnou news, mohou si mezi sebou předávat libovolné diskusní skupiny a mohou do nich přidávat své místní hierarchicky uspořádané skupiny. Například server **groucho.edu** může mít odkaz na server **bernard.edu**, který je hlavním zdrojem news, a několik odkazů na menší servery, které naopak zásobuje svými news. Takto může Bernyard College obdržet všechny usenetové skupiny, zatímco GMU má zájem jen o některé hlavní skupiny typu **sci**, **comp**, **rec** atd. Některé z následujících serverů, řekněme například UUCP-server nazvaný **brewhq**, bude mít zájem o ještě menší počet skupin, z důvodu nedostatku síťových nebo hardwarových zdrojů. Na druhou stranu může mít server **brewhq** zájem o diskusní skupiny z větve **fj**, které GMU nepřijímá. Pro takové případy udržuje jiný odkaz na **gargleblaster.com**, kde je umístěna celá hierarchie **fj** a ten předá serveru **brewhq**. Tok news ukazuje obrázek 16.1.



Obrázek 16.1
Tok usenetových news v Groucho Marx Univerzity

Popisky šipek vedoucích ze serveru **brewhq** vyžadují bližší vysvětlení. Implicitně chce totiž všechny místní news posílat na adresu **groucho.edu**. Avšak na tomto serveru nejsou uchovávány skupiny **fj** a neexistuje žádný ukazatel pro posílání jakýchkoliv zpráv z těchto skupin. Proto je předávání zpráv z **brewhq** do GMU označeno jako **all,!fj**, což znamená, že jsou univerzitě posílány všechny diskusní skupiny s výjimkou **fj**.

16.3 Jak Usenet obsluhuje news?

Dnes se Usenet rozrostl do obrovských rozměrů. Systémy, které uchovávají celé síťové news obvykle přenášejí něco kolem šedesáti megabajtů dat denně[2]. Samozřejmě, že to vyžaduje mnohem více než jen postrkování souborů. Podívejme se tedy na způsob, jakým většina unixových systémů obsluhuje usenetové news.

News jsou po síti šířeny různými druhy přenosů. Historickým médiem je UUCP, ale hlavní dopravu dnes obstarává Internet. Použitému směrovacímu algoritmu se říká flooding (zaplavování): každý systém udržuje několik spojení (přítoků news – news feeds) s jinými systémy. Každý článek vytvořený nebo obdržený lokálním systémem news jim takto předá, výjimku tvoří jen případ, kdy již příslušný článek existuje, a v takovém případě je tento článek zrušen. Systém může na základě hlavičkového pole `Path:` zjistit, kterými systémy již daný článek prošel. Tato hlavička obsahuje seznam všech systémů, které článek předaly dále, v podobě tzv. notace bang path.

Aby bylo možné články rozlišovat a odhalovat duplicity, nese si s sebou každý usenetový článek id zprávy (uvedené v hlavičkovém poli `Message-Id`), které kombinuje název odesílatele a sériové číslo do tvaru „<`sériové číslo@odesílatel`>“. Systém news uloží id každého článku, který zpracoval, do souboru `history`, s nímž jsou pak porovnávány všechny nově příchozí články.

Tok článků mezi dvěma místy může být omezen dvěma kritérii: zaprvé může být článku přiřazena tzv. distribuce (v hlavičkovém poli `Distribution:`), která může omezit jeho doručení jen do určitých skupin míst. Na druhé straně mohou být předávané skupiny news omezeny odesílatelem i příjemcem. Sada diskusních skupin a distribucí je obvykle uchovávána v souboru *sys*.

Velký počet článků zpravidla vyžaduje zdokonalení výše uvedeného schématu. V sítích UUCP je přirozené sesbírat články za určité období, zkombinovat je do jednoho souboru, ten potom zkomprimovat a poslat vzdálenému systému. Této proceduře říkáme *dávkování* (*batching*)[3].

Jinou alternativou je protokol *ihave/sendme*, který v prvé řadě zabraňuje přenosu duplicitních článků, čímž šetří i šířku pásma. Místo aby umístil do dávkových souborů všechny články a poslal je dál, zkombinuje pouze id zpráv článků do jedné velké zprávy „ihave“ a pošle ji vzdálenému systému. Ten si ji přečte, porovná její obsah s obsahem souboru history a v souboru „sendme“ vrátí seznam článků, které požaduje. Pak mu jsou poslány pouze tyto články.

2 Moment: 60 megabajtů při rychlosti 9600 bps, to je 60 milionů krát 1200, to je…šrum…šrum…, to je ale 34 hodin!

3 Zlaté pravidlo síťových news podle Geoffa Collyera zní: „Měl bys své články dávkovat.“

Samozřejmě, že protokol ihave/sendme má smysl jen u dvou velkých systémů, z nichž každý získává news z několika nezávislých zdrojů, které sčítají nové zprávy dostatečně často pro efektivní tok news.

Systémy, které jsou připojeny k Internetu, obecně spoléhají na software založený na protokolu TCP/IP, který využívá protokol NNTP (Network News Transfer Protocol)[4]. Zprostředkuje přenos news mezi zásobníky a zajišťuje jednotlivým uživatelům přístup ke vzdáleným hostitelům.

Protokol NNTP definuje tři různé způsoby přenosu news. Jedním z nich je verze ihave/sendme v reálném čase, které se také říká *tlačení* (*pushing*). Druhým způsobem je metoda *tažení* (*pulling*) news, při které klient požaduje seznam článků v dané diskusní skupině nebo hierarchii, které dorazily na příslušný server po specifikovaném datu, a vybere si ty, jež nenajde ve svém souboru history. Třetí metoda slouží k interaktivnímu čtení news a dovoluje vám nebo vašemu prohlížeči news získávat články ze specifikovaných diskusních skupin, stejně jako posílat články s neúplnými hlavičkovými informacemi.

Na každém serveru jsou news uchovávány v adresářové hierarchii `/var/spool/news`, každý článek je uložen v samostatném souboru a každá diskusní skupina má svůj vlastní adresář. Název adresáře je odvozen z názvu diskusní skupiny, přičemž jednotlivé komponenty tohoto názvu tvoří cestu. Takto budou články ze skupiny **comp.os.linux.misc** uloženy v adresáři `/var/spool/news/comp/os/linux/misc`. Článkům jsou v jednotlivých skupinách přidělována čísla podle pořadí, v jakém přišly. Toto číslo slouží jako název souboru. Rozsah čísel článků, které jsou aktuálně k dispozici, je uchováván v souboru *active*, který současně slouží jako seznam známých diskusních skupin ve vašem systému.

Jelikož je diskový prostor omezeným zdrojem[5], musí někdo po určité době články mazat. Tomuto mechanismu říkáme *vypršení platnosti* (*expiring*). Doba platnosti článků z určitých skupin a hierarchií zpravidla vyprší po pevném počtu dnů od jejich doručení. Toto pravidlo může odesílatel potlačit za pomoci pole `Expires:` v hlavičce článku, ve kterém uvede datum vypršení platnosti.

4 Jeho popis najdete v RFC 977.

5 Někteří lidé tvrdí, že Usenet je spiknutím dodavatelů modemů a pevných disků.

Kapitola

17

C News

Jedním z nejoblíbenějších softwarových balíků pro síťové news je balík C News. Byl navržen pro systémy, které přenášejí news prostřednictvím protokolu UUCP. Tato kapitola se zabývá základními vlastnostmi C News a základny instalace a údržby tohoto balíku.

Program C News ukládá své konfigurační soubory do adresáře `/usr/lib/news` a většina jeho binárních souborů je umístěna v adresáři `/usr/lib/news/bin`. Články jsou uchovávány pod adresářem `/var/spool/news`. Měli byste se ujistit, že všechny soubory v těchto adresářích jsou vlastněny uživatelem **news** ze skupiny **news**. Většina následujících problémů totiž souvisí s tím, že systém C News nemá přístup k příslušným souborům. Zvykněte si, že dříve než se čehokoliv v tomto adresáři dotknete se musíte přihlásit jako uživatel **news** prostřednictvím `su`. Jedinou výjimkou je program `setnewsids`, který slouží k nastavování skutečných uživatelských id některých programů news. Vlastníkem těchto programů musí být uživatel **root** a musí mít nastaven setuid bit.

V následujících statích si podrobně popíšeme všechny konfigurační soubory systému C News a řekneme si, co je třeba udělat pro to, aby váš systém správně fungoval.

17.1 Doručování news

Články lze do systému C News dodávat několika způsoby. Když pošle článek místní uživatel, program pro čtení news (newsreader) ho obvykle předá příkazu `inews`, který doplní hlavičkové informace. Články ze vzdálených systémů, ať už se jedná o jediný článek nebo o celou dávku článků, jsou předány příkazu `rnews`, který je uloží do adresáře `/var/spool/news/in.coming`, odkud si ho později vyzvedne program `newsrun`. Při použití obou výše zmíněných postupů však bude článek nakonec předán programu `relaynews`.

U každého článku příkaz `relaynews` nejprve vyhledá id zprávy v souboru `history`, čímž zjistí, zda již místní systém daný článek zná. Duplicitní články zruší. Potom se program `relaynews` podívá do hlavičky na řádek `Newsgroups:`, aby zjistil, zda daný systém přijímá články z některé z těchto skupin. Pokud ano a příslušná diskusní skupina je uvedena v souboru `active`, pokusí se program příslušný článek uložit do odpovídajícího adresáře v adresáři `spool`. Pokud takový adresář neexistuje, pak ho vytvoří. Id zprávy následně zaznamená do souboru `history`. V opačném případě program `relaynews` článek smaže.

Pokud se programu `relaynews` nepodaří uložit příchozí článek, protože skupina, do které byl zaslán, není uvedena v souboru `active`, umístí článek do skupiny **junk**.[1] Program *relaynews* také vyhledá staré články nebo články se špatným datem a zruší je. Příchozí dávky článků, jejichž zpracování z nějakého důvodu selže, umístí do adresáře `/var/spool/news/in.coming/bad` a do log-souboru zapíše chybu.

Poté je článek přesměrován na všechny ostatní systémy, které požadovaly news z těchto skupin, přičemž se využije přenos specifikovaný pro konkrétní místo. Aby nebyl článek posílán serveru, který ho již viděl, je každé místo určení porovnáno s hlavičkovým polem `Path:`, které obsahuje seznam míst (zapsaný v bang path style), přes která článek zatím prošel. Příslušný článek mu bude poslán pouze v případě, že se název cílového místa nevyskytuje v tomto seznamu.

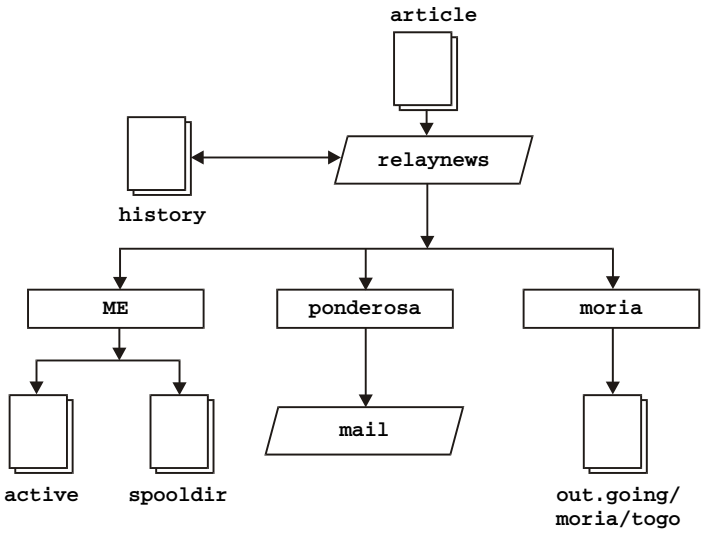
Systém C News je běžně používán k předávání news mezi systémy, které využívají protokol UUCP. Lze ho však používat i v prostředí NNTP. Pro doručování news (buď jednotlivých článků, nebo celých dávek) na vzdálený UUCP-systém se za pomoci `uux` spustí na vzdáleném systému program `rnews` a pošle článek nebo dávku na jeho standardní vstup.

Je-li pro daný systém povoleno dávkování, nepošle systém C News příchozí článek okamžitě, ale připojí název jeho cesty do souboru, který se většinou nazývá `out.going/system/togo`. Periodicky je ze záznamu `crontab` spouštěn program pro vytváření dávek[2], který umístí články do jednoho nebo více souborů, případně je i zkomprimuje a pošle je programu `rnews` na vzdáleném systému.

Tok news zprostředkovaný programem `relaynews` ukazuje obrázek 17.1. Články mohou být přenášeny do lokálního systému (označeného jako *ME*) a prostřednictvím elektronické pošty do systému nazvaného **panderosa** nebo místa nazvaného **moria**, pro které je zapnuto dávkování.

1 Mezi skupinami, které existují ve vašem systému a skupinami, o které máte zájem, mohou být značné rozdíly. Například seznam zapsaných skupin může uvádět skupinu **comp.all**, což znamená všechny skupiny pod hierarchií **comp**, ale vy máte v souboru `active` uvedeno jen několik skupin **comp**. Články zaslané do těchto skupin budou umístěny do skupiny **junk**.

2 Všimněte si, že by to měl být crontab systému **news**, aby nebyla porušena přístupová práva k souborům.



Obrázek 17.1
Tok news zprostředkovaný programem `relaynews`

17.2 Instalace

Při instalaci systému C News rozbalte soubory do příslušných adresářů (pokud jste tak již neučinili) a upravte níže uvedené konfigurační soubory. Všechny je najdete v adresáři `/usr/lib/news`. Jejich formáty si nyní popíšeme.

`sys` Pravděpodobně bude třeba upravit řádek *ME*, který popisuje váš systém, i když parametr *all/all* je bezpečný vždy. Doplňte do něj také řádek pro každý systém, kterému předáváte news.

Pokud jste koncovým uzlem, stačí vám jen řádek, který pošle všechny místně vytvořené články do vašeho zásobníku. Předpokládejme, že vaším dodavatelem je server **moria**, pak by váš soubor *sys* vypadal následovně:

```
ME:all/all::
moria/moria.orcnet.org:all/all,!local:f:
```

`organization` Název vaší organizace. Například „`Virtual Brewery, Inc`“. Na vašem domovském počítači zadejte něco na způsob „private site“. Většina lidí nebude žádat správnou konfiguraci vašeho systému, pokud tento soubor neupravíte.

`newsgroups`

`mailname` Poštovní adresa vašeho systému, tj. například **vbrew.com**.

`whoami` Název vašeho systému pro účely news. Často se používá UUCP-název místa, například **vbrew**.

`explist` Tento soubor byste asi měli upravit, protože definuje dobu platnosti některých speciálních diskusních skupin. Zde může hrát důležitou roli i disková kapacita.

Počáteční hierarchii vytvoříte tak, že si ze systému, který vás zásobuje novými news, obstaráte soubory `active` a `newsgroups` a nainstalujete je do adresáře `/usr/lib/news`. Také se ujistěte, že je jejich vlastníkem uživatel news a mají nastavena přístupová práva 644. Ze souboru `active` odstraňte všechny skupiny **to.***, **to**.*mysite* a **to**.*feedsite*, stejně jako **junk** a **control**. Skupiny **to.*** slouží k výměně zpráv protokolu ihave/sendme, ale měli byste je vytvořit bez ohledu na to, zda s použitím tohoto protokolu počítáte, či nikoli. Dále nahraďte pomocí následujícího příkazu všechna čísla článků ve druhém a třetím poli v souboru `active`.

```
# cp active active.old
# sed 's/ [0-9]* [0-9]* / 0000000000 00001 /' active.old > active
# rm active.old
```

Druhý příkaz, který je mým oblíbeným, spouští program `sed`*(1)*. Nahradí dva řetězce číslic řetězcem nul a řetězcem 000001.

Nakonec vytvoříme adresář `spool` a jeho podadresáře používané pro ukládání příchozích a odchozích news:

```
# cd /var/spool
# mkdir news news/in.coming news/out.going
# chown -R news.news news
# chmod -R 755 news
```

Používáte-li pozdější verzi systému C news, bude možná ještě nutné vytvořit v adresáři `spool` podadresář `out.master`.

Jestliže používáte pro čtení news jiný program, než jaký je dodáván se systémem C News, zjistíte, že váš adresář spool je místo v adresáři `/var/spool/news` vytvořen v adresáři `/usr/spool/news`. Nepodaří-li se vašemu programu pro čtení news najít příslušné články, vytvořte symbolický odkaz jménem `/usr/spool/news`, který bude ukazovat na adresář `/var/spool/news`.

Nyní jste připraveni na příjem news. Všimněte si, že už nemusíte vytvářet žádné jiné adresáře, ale pokaždé, když systém C News obdrží článek patřící do skupiny, pro kterou ještě neexistuje příslušný adresář, tento vytvoří.

Konkrétně to platí pro skupiny *alt*. Takže po nějaké době zjistíte, že váš adresář spool bude přeplněn adresáři s názvy skupin, které jste si nikdy nezapsali, například **alt.lang.teco**. Tomu zabráníte tak, že buď odstraníte ze souboru `active` všechny nechtěné skupiny, nebo budete pravidelně spouštět skript příkazového interpretu, který odstraní všechny prázdné adresáře pod adresářem `/var/spool/news` (vyjma `out.going` a `in.coming` samozřejmě).

Systém C News potřebuje účet, na který by mohl posílat chybové a stavové zprávy. Implicitně se tento účet nazývá **usenet**. Využijete-li toto implicitní nastavení, musíte nastavit alias, který by veškerou poštu došlou na tento účet přesměroval na odpovědné osoby. (Jak to provést u programů `smail` a `sendmail` popisujeme v 14. a 15. kapitole.) Toto chování lze také potlačit pomocí proměnné prostředí *NEWMASTER*, které přiřadíte příslušný název. Proměnnou je třeba nastavit v souboru crontab systému **news**, stejně jako při každém manuálním spuštění nějakého administrativního nástroje, takže zřízení aliasu je zřejmě jednodušší.

Při úpravě souboru `/etc/passwd` se ujistěte, že každý uživatel má v poli *pw_gecos* (jde o čtvrté pole) zapsáno svoje skutečné jméno. Patří k etiketě sítě (netiquette), že se v poli From: příslušného článku objeví skutečné jméno odesílatele. Stejně to musíte udělat, když budete používat elektronickou poštu.

17.3 Soubor sys

Soubor `sys`, umístěný v adresáři `/usr/lib/news`, řídí hierarchie skupin, které dostáváte, a jejich další směrování. I když existují speciální nástroje pro práci s tímto souborem, jmenovitě to jsou `addfeed` a `delfeed`, myslím, že je lepší upravovat tento soubor ručně.

Soubor *sys* obsahuje záznam pro každý systém, kterému předáváte news, stejně jako popis skupin, které přijímáte. Každý záznam má následující formát:

```
site[/exclusions]:grouplist[/distlist][:flags[:cmds]]
```

Jako pokračovací znak na novém řádku se používá znak zpětné lomítko. Znak (#) znamená komentář.

site Toto je název systému, kterého se záznam týká. Zpravidla se použije UUCP-název příslušného místa. V souboru *sys* musí mít záznam i váš systém, jinak byste nedostávali žádné články.

Speciální název *ME* označuje váš systém. Tento záznam definuje všechny skupiny, které chcete mít uloženy lokálně. Články, které nevyhovují řádku se záznamem *ME*, putují do skupiny **junk**.

Protože systém C News porovnává pole `site` s názvy systémů v hlavičkovém poli `Path:`, musí si tyto přesně odpovídat. Některé systémy předávají v tomto poli plně kvalifikované názvy domén nebo alias typu **news**.*systém.doména*. Aby nebyly těmto systémům vráceny žádné články, je třeba přidat tyto názvy do seznamu výjimek, ve kterém jsou jednotlivé položky odděleny čárkami.

V záznamu týkajícím se systému **moria** by mohlo příslušné pole obsahovat například řetězec **moria/moria.ocnet.org**.

grouplist Toto je seznam skupin a hierarchií oddělených čárkami, které si zapsal konkrétní systém. Hierarchii je možné specifikovat prostřednictvím předpony (například **comp.os** pro všechny skupiny, jejichž název začíná touto předponou), za níž může volitelně následovat klíčové slovo **all** (tj. **comp.os.all**).

Předání příslušné hierarchie nebo skupiny zabráníte tak, že před její název přidáte vykřičník. Při porovnávání názvu diskusní skupiny s tímto seznamem má nejvyšší prioritu nejdelší shoda. Když například pole *grouplist* obsahuje

```
!comp,comp.os.linux,comp.folklore.computers
```

nebudou příslušnému systému předány žádné skupiny z **comp** s výjimkou podskupin skupiny **comp.folklore.computers** a **comp.os.linux**.

Pokud daný systém požaduje všechny news, které jste sami obdrželi, zadejte do pole *grouplist* položku *all*.

distlist je offset oddělený od pole *grouplist* lomítkem a obsahující seznam distribucí, který se má předávat dále. I v tomto případě můžete některé distribuce vyřadit tím, že před ně umístíte znak vykřičník. Všechny distribuce označuje řetězec *all.* Pokud toto pole vynecháte, znamená to, že jste zvolili *all*.

Můžete například použít seznam distribucí *all,!local*, čímž zabráníte šíření news, které jsou určeny jen pro lokální použití.

Toto pole většinou obsahuje minimálně dvě distribuce: *world*, což je vždy implicitní distribuce, která se použije, když uživatel žádnou distribuci neuvede, a *local*. Existují také další distribuce, které se týkají určité oblasti, státu, země atd. Nakonec zde máme dvě distribuce, které používá pouze systém C News. Jsou to *sendme* a *ihave* a využívá je protokol ihave/sendme.

Využití distribucí je předmětem stálých sporů. Podle někoho vytváří některé programy pro čtení news falešné distribuce podle nejvyšší hierarchie, například **comp** při posílání news do skupiny **comp.os.linux**. Distribuce, které se týkají regionů, jsou také sporné, protože news, pokud je pošlete po Internetu, mohou cestovat mimo váš region.[3] Distribuce aplikované na organizace mají na druhé straně velký význam, zabraňují například šíření tajných informací mimo příslušnou organizaci. Tohoto cíle se však dosáhne lépe vytvořením samostatné diskusní skupiny nebo hierarchie.

flags Toto pole popisuje parametry přenosu. Může být prázdné nebo může obsahovat kombinaci následujících znaků:

- *F* Tento příznak povoluje dávkování.
- *f* Tento příznak je podobný výše uvedenému, ale umožňuje systému C News přesnější výpočet velikosti odcházejících dávek.
- *I* Tento příznak způsobí, že budou C News vytvářet seznam článků vhodný pro protokol ihave/sendme. Aby bylo možné tento protokol použít, jsou nutné další úpravy v souborech `sys` a `batchparams`.
- *n* Na základě tohoto příznaku budou vytvářeny dávkové soubory pro přenosové klienty využívající protokol NNTP, jako například `nntpxmit` (viz 18. kapitola). Dávkové soubory obsahují název souboru článku a id zprávy.
- *L* Tento příznak říká systému C News, aby přenášel pouze články zaslané vašemu systému. Příznak může být následován desítkovým číslem *n*, které způsobí, že budou C News přenášet pouze články, které dorazily během *n* hops. Systém C News určí počet hops z pole `Path:`.
- *u* Systém C News bude vytvářet dávky pouze z článků v neřízených (nemoderovaných) skupinách.
- *m* Systém C News bude vytvářet dávky pouze z článků v řízených skupinách.

Současně lze použít pouze jeden z příznaků *F*, *f*, *I* nebo *n*.

cmds Toto pole obsahuje příkaz, který se provede nad každým článkem, pokud není povoleno dávkování. Příslušný článek bude předán na standardní vstup tohoto příkazu. Tuto možnost byste měli využívat jen u malých dávek news, protože jinak by došlo k velkému zatížení obou systémů.

[3] Je zcela běžné, když článek poslaný řekněme z Hamburgu do Frankfurtu putuje přes uzel **reston.ans.net** v Holandsku, nebo dokonce přes některý uzel v USA.

Implicitní příkaz má formát

```
uux - -r -z system!rnews
```

a spustí na vzdáleném systému program `rnews`, jemuž předá na standardní vstup příslušný článek.

Implicitní vyhledávací cesta pro příkazy předané v tomto poli je `/bin:/usr/bin:/usr/lib/news/bin/batch`. Poslední z těchto adresářů obsahuje několik skriptů příkazového interpretu, jejichž názvy začínají slovem *via*. Podrobněji si je popíšeme dále v kapitole.

Je-li povoleno dávkování jedním z parametrů *F* nebo *f*, *I* nebo *n*, bude systém C News očekávat, že najde v tomto poli název souboru a ne příkaz. Pokud název souboru nezačíná lomítkem (/), předpokládá se, že jde o relativní cestu k adresáři `/var/spool/news/out.going`. Je-li toto pole prázdné, vezme se implicitní cesta *systém*`/togo`.

Při nastavování systému C News budete zřejmě muset vytvořit svůj vlastní soubor *sys*. Abychom vám to ulehčili, nabízíme vám vzorový soubor serveru **vbrew.com**, ze kterého si můžete zkopírovat, co potřebujete.

```
# Bereme vše, co nám ostatní poskytnou.
ME:all/all::

# Vše co dostaneme posíláme na moria kromě lokálních článků.
# Používáme dávkování.
moria/moria.orcnet.org:all,!to,to.moria/all,!local,!brewery:f:

# Skupinu comp.risks posíláme na adresu jack@ponderosa.uucp
ponderosa:comp.risks/all::rmail jack@ponderosa.uucp.

# Hostitel swim dostává několik skupin.
swim/swim.twobirds.com:comp.os.linux,rec.humor.oracle/all,!local:f:

# Zaznamenávej poštovní mapy pro pozdější zpracování.
usenet-maps:comp.mail.maps/all:F:/var/spool/uumaps/work/batch
```

17.4 Soubor active

Soubor `active` najdete v adresáři `/usr/lib/news`. Obsahuje seznam všech skupin, které zná váš systém, a všech článků, které jsou zde v současné době uloženy. Jen výjimečně ho budete přímo upravovat, ale kvůli úplnosti si ho zde probereme. Záznamy v tomto souboru mají následující formát:

```
newsgroup high low perm
```

Pole *newsgroup* je samozřejmě název skupiny. Pole *low* a high udávají nejnižší a nejvyšší čísla článků, které jsou právě dostupné. Není-li dostupný žádný článek, je pole *low* rovno high+1.

Přinejmenším to objasňuje účel pole low. Nicméně systém C News kvůli efektivitě toto pole neaktualizuje. To by nebyla tak velká ztráta, kdyby na něm nezávisely některé programy pro prohlížení news. Například program `trn` se na základě tohoto pole rozhoduje, zda může odstranit nějaké články ze své databáze vláken. Kvůli aktualizaci pole *low* je proto nutné pravidelně spouštět program `updatemin` (nebo v dřívějších verzích systému C News skript `upact`).

Parametr *perm* řídí přístup uživatelů, kteří mají povolen přístup k dané skupině. Může nabývat jedné z následujících hodnot:

y — Uživatelé mohou do této skupiny přispívat.

n — Uživatelé nemohou do této skupiny přispívat. Nicméně mohou z ní číst články.

x — Tato skupina byla lokálně znepřístupněna. K tomu občas dochází, když se administrátoři news (neboli superuživatelé) kvůli nějakému poslanému článku urazí.

Články určené pro tuto skupinu nejsou ukládány lokálně, nicméně jsou předávány dále systémům, které je požadují.

m — Tato hodnota označuje řízenou (moderovanou) skupinu. Pokusí-li se uživatel zaslat do této skupiny nějaký článek, inteligentní program pro čtení news ho na to upozorní a místo do skupiny ho pošle moderátorovi. Adresu moderátora zjistí ze souboru `moderators`, který najde v adresáři `/usr/lib/news`.

=real-group — Takto označíte skupinu *newsgroup* jako lokální alias (přezdívku) jiné skupiny, jmenovitě *real-group*. Všechny články zaslané do skupiny *newsgroup* budou přesměrovány do této skupiny.

V systému C News ve skutečnosti nebudete muset přímo pracovat s tímto souborem. Skupiny lze mazat lokálně i globálně pomocí nástrojů `addgroup` a `delgroup` (viz dále popis ve

stati Nástroje pro údržbu a úlohy). Je-li do Usenetu přidána nebo z něj naopak odstraněna nějaká skupina, dozví se o tom všechny systémy na základě řídicí zprávy *newgroup* nebo *rmgroup*. Nikdy však neposílejte tuto zprávu sami! Máte-li zájem o podrobnosti týkající se vytváření diskusních skupin, pak si přečtěte měsíční příspěvky do konference **news.announce.newusers**.

Se souborem `active` je úzce spjat soubor `active.times`. Kdykoliv je vytvořena nová skupina, zaznamená systém C News do tohoto souboru zprávu, která obsahuje název vytvořené skupiny, datum vytvoření, zda se tak stalo na základě zprávy *newgroup* nebo lokálně a kdo tak učinil. To kvůli programům pro čtení news, které tak mohou informovat uživatele o každé nově vytvořené skupině. Tuto informaci využívá také příkaz *NEWGROUPS* protokolu NNTP.

17.5 Dávkování článků

Dávky news dodržují konkrétní formát, který je stejný pro Bnews, C News a INN. Každému článku předchází následující řádek:

```
#! rnews count
```

kde *count* je počet bajtů článku. Použije-li se dávková komprese, je výsledný soubor zkomprimován jako celek a na začátek je umístěn další řádek, který udává příkaz, jež se použije pro rozbalení. Standardním kompresním nástrojem je `compress`, který je označen zprávou

```
#! cunbatch
```

Někdy, když je potřeba poslat dávky za pomocí poštovního programu, který ze všech dat odstraňuje osmý bit, je možné zkomprimovanou dávku chránit prostřednictvím tzv. c7-kódování. Takovéto dávky budou označeny `c7unbatch`.

Když je dávka na vzdáleném systému předána programu `rnews`, vyhledá tyto značky a dávku příslušným způsobem zpracuje. Některé systémy používají ještě jiné kompresní nástroje, například `gzip`, a před takto zkomprimované soubory předřazují zprávu `zunbatch`. Systém C News těmto nestandardním hlavičkám nerozumí. Má-li je podporovat, je třeba upravit zdrojový kód.

V systému C News je dávkování článků prováděno programem `/usr/lib/news/bin/batch/sendbatches`, který vezme seznam článků ze souboru `system/togo` a umístí je do několika dávek. Měl by být spouštěn jednou za hodinu případně ještě častěji, v závislosti na objemu provozu.

Jeho práci řídí soubor `batchparams` umístěný v adresáři `/usr/lib/news`. Tento soubor popisuje maximální velikost dávky pro každý systém, dávkovací a volitelně i komprimační

program, který se má použít, a přenos, který zajistí doručení příslušné dávky vzdálenému systému. Dávkovací parametry je možné zadat pro každý systém jednotlivě nebo jako sadu implicitních parametrů, kde není explicitně zmiňován žádný systém.

Dávkování pro specifický systém spustíte následujícím příkazem

```
# su news -c "/usr/lib/news/bin/batch/sendbatches site"
```

Spustíte-li program *sendbatches* bez argumentů, bude obsluhovat veškeré dávkovací dotazy. Interpretace slova „all" závisí na přítomnosti implicitního záznamu v souboru `batchparams`. Pokud existuje, budou se kontrolovat všechny adresáře uvedené v adresáři `/var/spool/news/out.going`. V opačném případě bude program procházet všechny záznamy uvedené v souboru `batchparams`. Všimněte si, že program *sendbatches* bere při procházení adresáře `out.batches` v úvahu pouze adresáře, které nemají v názvu tečku nebo znak at (@).

Při instalaci systému C News pravděpodobně zjistíte, že součástí distribuce je i soubor `batchparams`, který obsahuje rozumné implicitní záznamy, takže zřejmě tento soubor nebudete muset upravovat. Jeho formát si zde popíšeme jen pro úplnost. Každý řádek je složen ze šesti polí, které jsou navzájem odděleny mezerami nebo tabulátory:

```
site size max batcher muncher transport
```

Následuje popis významu těchto polí:

Pole *site* udává název systému, kterého se daný záznam týká. Soubor `togo` musí být v tomto systému umístěn v adresáři `spool` jako `out.going/togo`. Název místa `/default/` označuje implicitní záznam.

Pole *size* udává maximální velikost vytvořených dávek článků (před kompresí). Je-li velikost jednoho článku větší než tento údaj, povolí systém C News výjimku a umístí ho do samostatné dávky.

Pole max udává maximální počet dávek vytvořených a naplánovaných pro přenos, než je dávkování pro tento konkrétní systém zrušeno. To je užitečné v případě, že vzdálený systém dlouhou dobu nefunguje, protože se tak vyhnete zahlcení UUCP-adresářů `spool` miliony dávek news.

Systém C News určuje velikost čekajících dávek za pomoci skriptu `queulen`, který najdete v adresáři `/usr/lib/news/bin`. Balík `newspak`, jehož autorem je Vince Skahan, by měl obsahovat skript pro UUCP kompatibilní s BNU. Používáte-li jiné uspořádání adresáře spool, například Taylor UUCP, budete si možná muset napsat svůj vlastní skript.[4]

[4] Nezajímá-li vás počet souborů v adresáři `spool` (protože jste jediná osoba, která používá daný počítač, a nepíšete články o velikostech řádově megabajtů), můžete obsah tohoto skriptu nahradit jednoduchým příkazem `exit 0`.

Pole *batcher* obsahuje příkaz, který slouží k vytváření dávky ze seznamu článků v souboru `togo`. Pro účely normálního přenosu je jím zpravidla program `batcher`. Pro jiné účely mohou být poskytovány jiné dávkovací programy. Například protokol ihave/sendme vyžaduje, aby byl seznam článků převeden do řídicích zpráv protokolu ihave nebo sendme, které jsou pak odeslány do skupiny **to**.`site`. To mají na starosti programy `batchih` a `batchsm`.

Pole *muncher* specifikuje příkaz, který se použije při kompresi. Zpravidla to bývá skript **compcun**, který vytvoří komprimovanou dávku.[5] Pro stejné účely je možné využít také programy `gzip`, řekněme `gzipcun` (aby bylo jasno, musíte si ho sami napsat). Je třeba se ujistit, že program `uncompress` na vzdáleném systému je příslušným způsobem upraven, aby rozpoznal soubory komprimované programem `gzip`.

Pakliže vzdálený systém nedisponuje příkazem `uncompress`, stačí použít volbu `ncomp`, která kompresi zakáže.

Poslední pole *transport* popisuje přenos, který se má použít. K dispozici je několik příkazů, jejichž názvy začínají řetězcem *via* a slouží pro různé druhy přenosů. Program `sendbatches` jim předá na příkazové řádce název cílového systému. Pokud jste nenastavili hodnotu `batchparams` na */default/*, odvodí název systému z pole *site*, přičemž odstraní veškeré znaky za první tečkou nebo lomítkem včetně. V případě položky */default/* se použijí názvy podadresářů adresáře `out.going`.

Program `uux` používá ke spouštění `rnews` dva příkazy: `viauux` a `vaiuuxz`. Druhý z nich nastavuje příznak `-z` pro (starší verzi) programu `uux`, který způsobí, že tento program nebude vracet zprávy o úspěšném doručení každého článku. Jiný příkaz, `viamail`, posílá dávky článků uživateli **rnews** na vzdáleném systému prostřednictvím elektronické pošty. Samozřejmě, že tento vzdálený systém musí veškerou poštu pro uživatele **rnews** přesměrovat do svého lokálního systému news. Úplný seznam všech přenosů naleznete v manuálových stránkách příkazu `newsbatch`*(8)*.

Všechny příkazy z posledních tří polí musí být umístěny v adresáři `out.going`*/systém* nebo `/usr/lib/news/bin/batch`. Většina z nich jsou skripty, takže si můžete nové nástroje snadno přizpůsobit svým potřebám. Jsou spouštěny jako roura. Seznam článků je předáván na standardní vstup dávkovacího programu, jehož výstupem je příslušná dávka. Ta je předána programu na zpracování dávek atd.

[5] Skript `compcun`, který je dodáván jako součást balíku C News, používá program `compress` s 12bitovou kompresí, protože to je standard většiny systémů. Můžete vytvořit kopii tohoto souboru, řekněme `compcun16`, který by používal 16bitovou kompresi. Toto vylepšení však není tak působivé.

Následuje ukázkový soubor:

```
# soubor batchparms
# site      | size     |max   |batcher    |muncher    |transport
#-------------+--------+------+---------+-----------+-----------
/default/     100000     22  batcher       compcun    viauux
swim           10000     10  batcher       nocomp     viauux
```

17.6 Vypršení platnosti news

V systému Bnews se o aktuálnost news stará program nazvaný `expire`, kterému předáte jako argumenty seznam diskusních skupin a časový údaj, po jehož uplynutí vyprší platnost článků. Pokud chcete, aby různým hierarchiím vypršela platnost v jinou dobu, potřebujete skript, který by pro každou skupinu spouštěl program `expire` samostatně. Systém C News nabízí pohodlnější řešení: v souboru `explist` stačí uvést diskusní skupiny a příslušné časové intervaly. Jednou denně je z `cron` spouštěn příkaz `doexpire`, který podle tohoto seznamu zpracuje všechny skupiny.

Někdy si možná budete chtít ponechat články z určitých skupin i po uplynutí doby jejich platnosti; například si budete chtít archivovat programy zaslané do skupiny **comp.sources.unix**. Soubor `explist` umožňuje označit skupiny, které chcete takto archivovat.

Záznam v souboru `explist` vypadá následovně:

```
grouplist perm times archive
```

Pole *grouplist* je čárkami oddělený seznam diskusních skupin, kterých se daný záznam týká. Hierarchii skupin zadáte celým názvem skupiny, za nímž může nepovinně následovat řetězec *all*. Například záznam týkající se všech podskupin skupiny **comp.os** můžete v poli *grouplist* uvést jako **comp.so** nebo **comp.os.all**.

Při zjišťování vypršení platnosti news z nějaké skupiny je její název v příslušném pořadí porovnán se všemi záznamy v souboru `explist`. Použije se první vyhovující záznam. Chcete-li například po čtyřech dnech zrušit většinu obsahu diskusní skupiny **comp** s výjimkou skupiny **comp.os.linux.announce**, jejíž obsah si chcete ponechat týden, vytvoříte záznam pro druhou skupinu, který bude udávat sedmidenní platnost článků v ní, a za ním bude následovat záznam pro skupinu **comp**, který bude udávat platnost pouze čtyři dny.

Pole *perm* popisuje, zda se příslušný záznam týká moderovaných, nemoderovaných nebo všech skupin. Může nabývat hodnot *m*, *u* nebo *x*, které označují moderované, nemoderované a všechny skupiny.

Třetí pole, *times*, většinou obsahuje pouze jediné číslo. To je počet dnů, po jejichž uplynutí vyprší doba platnosti článků, které ji neměly explicitně nastavenu v poli `Expires:` v hlavičce článku. Všimněte si, že se jedná o počet dnů od data, kdy článek dorazil do vašeho systému, nikoliv od data jeho odeslání.

Pole *times* ovšem může být i složitější. Může obsahovat kombinaci až tří čísel, navzájem oddělených pomlčkou. První určuje počet dnů, které musí uplynout, aby se článek stal kandidátem na vyřazení. Jen zřídka se používá jiná hodnota než nulová. Druhé pole je výše zmiňovaný implicitní počet dnů, po kterém vyprší platnost souboru. Třetí číslo specifikuje počet dnů, po kterých bude článek bezpodmínečně smazán, bez ohledu na to, zda má v hlavičce uvedeno pole `Expires:`. Uvedete-li pouze prostřední číslo, použijí se pro zbylé dvě pole implicitní hodnoty. Ty je možné specifikovat prostřednictvím speciálního záznamu */bounds/*, který bude popsán dále.

Čtvrté pole, *archive*, určuje, zda má být příslušná diskusní skupina archivována, či nikoli. Pokud si archivaci nepřejete, pak zde uveďte pomlčku. V opačném případě zadejte buď úplnou cestu (odkazující na adresář), nebo znak at (@). Ten zastupuje implicitní archivní adresář, který pak musíte programu `doexpire` předat pomocí argumentu `-a` v příkazové řádce. Archivní adresář by měl vlastnit uživatel **news**. Když program `doexpire` archivuje článek řekněme ze skupiny **comp.sources.unix**, uloží ho do adresáře `comp/sources/unix` pod archivním adresářem a pokud neexistuje, vytvoří nový. Vlastní archivní adresář ovšem nevytvoří.

Program `doexpire` spoléhá na dva speciální záznamy ze souboru `explist`. Místo názvů diskusních skupin používají klíčová slova */bounds/* a */expired/*. Záznam */bounds/* obsahuje implicitní hodnoty tří položek pole *times*, které jsme si před chvílí popsali.

Pole */expired/* určuje, jak dlouho bude systém C News uchovávat řádky v souboru `history`. Tato položka je nutná z toho důvodu, že systém C News neodstraní řádek s názvem příslušného článku ze souboru `history` ihned po jeho smazání, ale ponechá ho tam pro případ, že by po tomto datu přišel duplicitní článek. Pokud dostáváte články pouze z jediného systému, může být tato hodnota malá. V ostatních případech se pro sítě založené na protokolu UUCP doporučují použít hodnotu odpovídající dvěma týdnům, v závislosti na zkušenostech s prodlevami článků z těchto systémů.

Vzorový soubor `explist` s krátkými intervaly vypršení platnosti zde reprodukujeme:

```
# keep history lines for two weeks. Nobody gets more than three month.
/expired/                          x         14        -
/bounds/                                     0-1-90    -

# groups we want to keep longer than the rest
comp.os.linux.announce             m         10        -
```

```
comp.os.linux                   x       5       -
alt.folklore.computers          u       10      -
rec.humor.oracle                m       10      -
soc.feminism                    m       10      -

# Archive *.sources groups
comp.sources,alt.sources        x       5       @

# defaults for tech groups
comp,sci                        x       7       -

# enough for a long weekend
misc,talk                       x       4       -

# throw away junk quickly
junk                            x       1       -

# Archive *.sources groups
comp.sources,alt.sources        x       5       @

# defaults for tech groups
comp,sci                        x       7       -

# enough for a long weekend
misc,talk                       x       4       -

# throw away junk quickly
junk                            x       1       -

# control messages are of scant interest, too
control                         x       1       -

# catch-all entry for the rest of it
all                             x       2       -
```

V souvislosti s vypršením vyvstává několik potenciálních problémů. Váš program pro čtení news může například spoléhat na třetí pole aktivního souboru, které obsahuje číslo nejnižšího článku, který je k dispozici. Při vyřazování článků systém C News toto pole neaktualizuje. Pokud potřebujete (nebo chcete), aby toto pole odráželo aktuální stav, pak musíte po každém spuštění programu `doexpire`[6] spustit i program `updatemin`.

Systém C News dále neprovádí kontrolu platnosti souborů procházením adresáře příslušné diskusní skupiny, ale pouze na základě souboru `history`.[7] Budou-li v souboru history nesprávné údaje, mohou články zůstat na vašem systému navždy, protože C News na ně prostě zapomenou.[8] Tento problém lze opravit pomocí skriptu `addmissing`, který najdete v adresáři `/usr/lib/news/bin/maint` a který přidá do souboru `history` chybějící články. Jiný skript, `mkhistory`, znovu vytvoří celý soubor `history`. Dříve než tento skript spustíte se nezapomeňte přihlásit jako uživatel **news**, jinak získáte soubor `history`, který bude pro systém C News nečitelný.

17.7 Různé soubory

Chování systému C News řídí více souborů, ale jejich existence není pro jeho fungování životně důležitá. Všechny sídlí v adresáři `/usr/lib/news`. Nyní si je krátce popíšeme.

`newsgroups` Tento soubor je společníkem souboru `active` a obsahuje seznam názvů diskusních skupin společně s popisem jejich hlavního tématu. Tento soubor je automaticky aktualizován, jakmile systém C News obdrží řídicí zprávu *checknews* (viz stať 17.8).

`localgroups` Máte-li více lokálních skupin, na které nechcete, aby vás systém C News upozorňoval pokaždé, když obdrží zprávu *checknews*, umístěte jejich názvy a popisy do tohoto souboru stejným způsobem, jak byste je zařadili do souboru `newsgroups`.

`mailpath` Tento soubor obsahuje adresu moderátorů všech moderovaných skupin. Každý řádek obsahuje název skupiny následovaný poštovní adresou moderátora (odsazenou tabulátorem).

6 U starších verzí systému C News to měl na starosti skript `upact`.

7 Datum, kdy článek dorazil, je uchováváno v prostředním poli na řádku v souboru history a je vyjádřeno v sekundách od 1. ledna 1970.

8 Nevím, *proč* k tomu dochází, ale na mém počítači se to občas stává.

Implicitní jsou dva speciální záznamy – *backbone* a *internet*. Oba označují (v notaci bang-path) cestu k nejbližšímu páteřnímu místu a systému, který rozumí adrese podle standardu RFC 822 (**uživatel@hostitel**). Implicitními záznamy jsou

```
internet      backbone
```

Máte-li nainstalován jeden z programů `smail` nebo `sendmail`, pak nemusíte záznam *internet* měnit, protože tyto programy rozumí adresování podle standardu RFC.

Záznam *backbone* se využije, když uživatel přispívá do moderované skupiny, jejíž moderátor není explicitně uveden. Je-li například název skupiny **alt.sewer** a záznam *backbone* obsahuje řetězec `path!%s`, pošle systém C News tento článek na adresu `path`**!alt-sewer** doufaje, že počítač backbone bude schopen článek předat dál. Jakou máte použít cestu zjistíte od správců news systému, který vás jimi zásobuje. Jako poslední útočiště můžete také použít adresu **uunet.uu.net!%s**.

`distributions`

Tento soubor ve skutečnosti nepatří systému C News, ale používají ho některé programy pro čtení news a `nntpd`. Obsahuje seznam distribucí, které rozpozná váš systém, a popis jejich (zamýšleného) efektu. Například společnost Virtual Brewery používá soubor, který obsahuje následující záznamy:

```
world      everywhere in the world
local      Only local to this site
nl         Netherlands only
mugnet     MUGNET only
fr         France only
de         Germany only
brewery    Virtual Brewery only
```

`log` Tento soubor obsahuje záznam všech aktivit systému C News. Pravidelně ho vybírá program `newsdaily`. Kopie starších log-souborů jsou pojmenovávány `log.o`, `log.oo` atd.

`errlog` Do tohoto souboru jsou zaznamenávány všechny chybové zprávy vyprodukované systémem C News. To se netýká článků, které byly odhozeny kvůli špatné skupině apod. Když program `newsdaily` zjistí, že je tento soubor neprázdný, automaticky ho pošle správci news (implicitně na účet **usenet**).

Soubor `errlog` vyprazdňuje program `newsdaily`. Starší kopie tohoto souboru jsou uchovávány pod názvy typu `errlog.o`.

`batchlog` Sem jsou zaznamenávána veškerá spuštění programu *sandbatches*. Tento soubor má zpravidla jen malý význam. Navštěvuje ho také program `newsdaily`.

`watchtime` Jedná se o prázdný soubor, který se vytvoří vždy při spuštění programu `newsmatch`.

17.8 Řídicí zprávy

Protokol usenetových news rozeznává speciální kategorii článků, které ze strany systému news evokují jisté odpovědi nebo akce. Takovýmto článkům říkáme *řídicí zprávy*. Poznáte je podle přítomnosti pole `Control:` v hlavičce článku, které obsahuje název řídicí operace, která se má provést. Existuje jich několik typů a všechny jsou obsluhovány skripty umístěnými v adresáři `/usr/lib/news/ctl`.

Většina z nich provede příslušnou akci automaticky v době, kdy je článek zpracován systémem C News, aniž by o tom informovaly správce news. Implicitně jsou správci předány pouze zprávy *checkgroups*, ale toto chování lze úpravou příslušných skriptů změnit.

17.8.1 Zpráva cancel

Nejznámějším typem zprávy je zpráva *cancel*, s jejíž pomocí může uživatel zrušit článek, který dříve poslal. Pakliže tento článek dosud existuje, zajistí tato zpráva jeho efektivní odstranění z adresářů *spool*. Zpráva *cancel* je předána všem systémům, které dostávají news z příslušné skupiny, bez ohledu na to, zda příslušný článek již viděli či nikoli. To pro případ, že by se původní článek opozdil za rušící zprávou. Některé systémy news dovolují uživatelům rušit zprávy jiných osob; to je samozřejmě definitivní ne-ne.

17.8.2 Zprávy newgroup a rmgroup

Tyto dvě zprávy se týkají vytváření a rušení diskusních skupin. Nová skupina může být v „obvyklé“ hierarchii vytvořena až poté, co to odsouhlasí uživatelé Usenetu. Pravidla týkající se hierarchie skupiny **alt** dovolují vzniknout něčemu, co se hodně podobá anarchii. Více podrobností najdete ve skupinách **news.announce.newusers** a **news.announce.newgroups**. Nikdy neposílejte zprávu *newgroup* nebo *rmgroup*, pokud bezpodmínečně nevíte, že k tomu máte oprávnění.

17.8.3 Zpráva checkgroups

Zprávy *checkgroups* posílají správci news, aby všechny systémy v síti provedly vzájemnou synchronizaci svých souborů `active` s realitou Usenetu. Komerční poskytovatelé internetových služeb by mohli například poslat tuto zprávu svým zákazníkům. Jednou měsíčně pošle moderátor skupiny **comp.announce.newgroup** „oficiální“ zprávu *checkgroups* pro hlavní hierarchie. Tato zpráva je ovšem poslána jako běžný článek, nikoliv jako řídicí zpráva. Chcete-li provést operaci *checkgroup*, uložte tento článek do souboru, řekněme `/tmp/check`, odstraňte z něj vše kromě vlastní řídicí zprávy a předejte ho následujícím způsobem skriptu *checkgroups*:

```
#   su   news -c "/usr/lib/news/bin/ctl/checkgroups"  < /tmp/check
```

Tento příkaz aktualizuje váš soubor `newsgroups`, do kterého přidá skupiny uvedené v souboru `localgroups`. Původní soubor `newsgroups` bude přejmenován na `newsgroups.bac`. Všimněte si, že pošlete-li tuto zprávu lokálně, bude jen zřídka fungovat, protože program `inews` odmítne tak velký článek přijmout.

Když systém C News zjistí rozpory mezi seznamem *checkgroups* a souborem `active`, vytvoří seznam příkazů, které zajistí aktualizaci vašeho systému, a pošle ho správci news. Výstup má zpravidla následující podobu:

```
From news Sun Jan 30 16:18:11 1994
Date: Sun, 30 Jan 94 16:18 MET
From: news (News Subsystem)
To: usenet
Subject: Problems with your active file

The following newsgroups are not valid and should be removed.
alt.ascii-art
bionet.molbio.gene-org
comp.windows.x.intrisics
de.answers

You can do this by executing the commands:
/usr/lib/news/bin/maint/delgroup alt.ascii-art
/usr/lib/news/bin/maint/delgroup bionet.molbio.gene-org
/usr/lib/news/bin/maint/delgroup comp.windows.x.intrisics
/usr/lib/news/bin/maint/delgroup de.answers
```

```
The following newsgroups were missing.
comp.binaries.cbm
comp.databases.rdb
comp.os.geos
comp.os.qnx
comp.unix.user-friendly
misc.legal.moderated
news.newsites
soc.culture.scientists
talk.politics.crypto
talk.politics.tibet
```

Pokud od systému news obdržíte zprávu tohoto typu, pak jí slepě nevěřte. V závislosti na tom, kdo poslal zprávy *checkgroups*, může postrádat několik skupin až celé hierarchie. Skupiny byste tedy měli odstraňovat jen opatrně. Pokud zjistíte, že některé skupiny, jež chcete vést ve vašem systému, chybí, pak je musíte doplnit pomocí skriptu `addgroup`. Uložte seznam chybějících skupin do souboru a předejte ho následujícímu malému skriptu:

```
#!/bin/sh
cd /usr/lib/news

while read group; do
if grep -si "^$group[[:space:]].*moderated" newsgroup; then
mod=m
else
mod=y
fi
/usr/lib/news/bin/maint/addgroup $group $mod
done
```

17.8.4 Zprávy sendsys, version a senduuname

Nakonec zde máme tři zprávy, které lze využít k získávání informací o síťové topologii. Jsou to zprávy *sendsys*, *version* a *senduuname*. Systém C News pošle po jejich obdržení odesílateli buď soubor `sys`, řetězec s verzí softwaru nebo výstup programu `uuname`*(1)*. C News jsou při poskytování zpráv ohledně verzí velmi skoupé; vrátí pouze písmeno „C“.

I zde platí, že tento typ zpráv byste neměli posílat, pokud si nejste absolutně jistí, že nemohou opustit vaši (regionální) síť. Odpovědi na zprávy *sendsys* mohou snadno shodit síť UUCP.[10]

17.9 C News v prostředí NFS

Jednoduchý způsob, jak šířit news v rámci lokální počítačové sítě, je uchovávat všechny news na ústředním hostiteli a relevantní soubory exportovat prostřednictvím souborového systému NFS, kdy programy pro čtení news mohou přímo procházet články. Výhodou této metody oproti využití protokolu NNTP je mnohem nižší režie spojená se získáváním a řazením článků. Protokol NNTP na druhé straně vítězí v heterogenní síti, kde se vybavení jednotlivých hostitelů výrazně liší nebo kde uživatelé nemají ekvivalentní účty na serveru.

Při použití souborového systému NFS musí být články zaslané lokálnímu hostiteli přesměrovány na centrální počítač, protože přístup k administrativním souborům by v opačném případě mohl vystavit systém nebezpečným podmínkám, které by způsobily nekonzistenci souborů. Nebo se můžete rozhodnout chránit vaši oblast spool tím, že ji exportujete jen pro čtení, což také vyžaduje přesměrování na centrální počítač.

Systém C News provádí tyto úkoly transparentně. Jakmile pošlete nějaký článek, váš program pro čtení news zpravidla spustí program `inews`, který vloží daný článek do systému news. Tento příkaz podrobí článek několika kontrolám, doplní hlavičku a projde soubor `server` v adresáři `/usr/lib/news`. Pokud tento soubor existuje a obsahuje jiný název hostitele, než je název lokálního hostitele, spustí na tomto serveru prostřednictvím `rsh` program `inews`. Protože tento skript používá několik binárních příkazů a podporuje soubory systému C News, musíte mít buď lokálně nainstalován C News, nebo si připojit software news ze serveru.

Aby spojení `rsh` správně fungovalo, musí mít každý uživatel na serveru ekvivalentní účet, tj. takový, na který se může přihlásit bez hesla.

Ujistěte se, že název hostitele v souboru `server` přesně odpovídá výstupu příkazu `hostname`*(1)* na serveru. Jinak by se totiž systém C News při doručování článku navždy zacyklil.

17.10 Nástroje pro údržbu

Navzdory složitosti systému C News může být život jeho správce poměrně snadný, protože tento systém nabízí širokou paletu nástrojů pro údržbu. Některé z nich jsou určeny pro pravidelné spouštění z `cron`, například `newsdaily`. Používání těchto skriptů významně snižuje nároky na denní péči o vaši instalaci C News.

10 Ani na Internetu bych to nezkoušel.

Nebude-li uvedeno jinak, jsou následující příkazy umístěny v adresáři `/usr/lib/news/bin/maint`. Nezapomeňte, že před spuštěním těchto příkazů, se musíte přihlásit jako uživatel **news**. Pokud byste tyto příkazy spustili jako superuživatel, mohlo by dojít k tomu, že se tyto soubory stanou pro systém C News nepřístupnými.

`newsdaily` Význam tohoto souboru je zřejmý z jeho názvu. Jde o důležitý skript, který pomáhá udržet log-soubory přijatelně velké a ponechává poslední tři kopie každého z nich. Také se pokouší napravit různé anomálie, jako jsou staré dávky v příchozích a odchozích adresářích, příspěvky do neznámých nebo moderovaných diskusních skupin atd. Výsledné chybové zprávy pak pošle správci news.

`newswatch` Tento skript byste měli spouštět pravidelně, zhruba jednou za hodinu, protože hledá anomálie v systému news. Má vystopovat problémy, které by měly okamžitý vliv na funkčnost systému news a poslat o tom správci hlášení. Mezi kontrolované oblasti patří zamykací soubory, které se nepodařilo odstranit, neobsloužené vstupní dávky a nedostatek místa na disku.

`addgroup` Přidá do vašeho lokálního systému novou skupinu. Správný formát tohoto příkazu je

```
addgroup groupname y|n|m|=realgroup
```

Druhý argument má stejný význam, jako příznak v souboru `active` a může znamenat, že do dané skupiny může přispívat kdokoliv (g), nikdo (n), že je moderovaná (m) nebo že se jedná o alias pro jinou skupinu (*=realgroup*).

Tento příkaz možná využijete také v případě, kdy první články nově vytvořené skupiny dorazí dříve, než řídicí příkaz *newgroup*, který má vyvolat její vytvoření.

`delgroup` Umožňuje lokální smazání nějaké skupiny. Spustíte ho jako

```
delgroup groupname
```

Pořád ještě ale musíte smazat články, které zůstaly v adresáři spool. Anebo můžete jejich odstranění svěřit přirozenému běhu věcí (tj. programu *expire*).

`addmissing` Přidá chybějící články do souboru `history`. Tento skript použijte, máte-li podezření, že se nějaké články v systému zasekly.[11]

[11] Vždycky mě zajímalo, jak se zbavit článku „Pomoc, nemůžu rozchodit X11 s 0.97.2!!!“.

`newsboot` Tento skript by se měl spouštět při zavádění systému. Odstraní všechny zamykací soubory, které zbyly po „zabití“ procesů news při ukončování systému a uzavře a provede všechny dávky, které zde zůstaly z NNTP-spojení, jež byly přerušeny při ukončování systému.

`newsrunning` Tento skript najdete v adresáři `/usr/lib/news/bin/input` a lze ho použít k zakázání rozbalování příchozích news, například v pracovní době. Rozbalování dávek vypnete následujícím způsobem

```
/usr/lib/news/bin/input/newsrunning off
```

Rozbalování znovu zapnete, když místo řetězce *off* použijete *on*.

Kapitola

18

Popis protokolu NNTP

18.1 Úvod

Z důvodu existence různých přenosových protokolů poskytuje protokol NNTP zcela odlišný způsob výměny news, než jaký používá systém C News. NNTP znamená „Network News Transfer Protocol“ (Protokol pro přenos síťových news) a není vlastním softwarovým balíkem, nýbrž internetovým standardem[1]. Je založen na „proudově orientovaném“ spojení – většinou prostřednictvím protokolu TCP – mezi klientem na libovolném místě v síti a serverem, který uchovává síťové news ve svém diskovém prostoru. Toto proudové spojení umožňuje klientovi a serveru interaktivní přenos článků s takřka nulovým zpožděním, čímž se daří udržovat nízký počet duplicitních článků. Společně s vysokými přenosovými rychlostmi Internetu se dosahuje přenosu news, který daleko překonává původní sítě UUCP. Zatímco před dvěma roky bylo zcela běžné, když článek dorazil do posledního uzlu Usenetu za dva týdny, podařilo se nyní tuto dobu zkrátit na méně než dva dny a v samotném Internetu je to často otázka několika minut.

K získávání, posílání a předávání článku slouží klientům různé příkazy. Rozdíl mezi odesláním a předáním je ten, že druhá metoda se může týkat článků s neúplnými hlavičkovými informacemi[2]. Nalezení článku mohou využívat klienti pro přenos news stejně jako programy pro čtení news. Tím se stává protokol NNTP vynikajícím nástrojem, který umožňuje mnoha klientům v lokální síti přístup k news, přičemž se vyhnou zkomoleninám, k nimž dochází při použití NFS.

---

1 Formálně je specifikován v RFC 977.

2 Při předávání článku prostřednictvím protokolu NNTP k němu server vždy přidá alespoň jedno hlavičkové pole, které se jmenuje `Nntp-Posting-Host:`. Obsahuje hostitelský název klienta.

Protokol NNTP umožňuje také aktivní a pasivní způsob přenosu news, kterému se hovorově říká „pushing“ (tlačení) a „pulling“ (tažení). Tlačení používá v podstatě stejný způsob, jako protokol C News ihave/sendme. Klient nabízí serveru články prostřednictvím příkazu `IHAVE<varmsig>` a server mu vrátí odpověď, z níž vyplývá, zda již daný článek má, nebo ho požaduje. Pokud o něj má zájem, klient mu článek pošle a ukončí ho tečkou na samostatném řádku.

Tlačení news má jednu nevýhodu v tom, že poměrně výrazně zatěžuje systém serveru, protože ten musí kvůli každému článku prohledávat databázi historie.

Opačnou technikou je tažení news, kdy klient požaduje seznam všech (dostupných) článků diskusní skupiny, které dorazily po specifikovaném datu. Takovýto dotaz předává pomocí příkazu `NEWNEWS`. Ze získaného seznamu id-zpráv si klient vybere ty články, které ještě nevlastní a postupně o ně požádá server příkazem *ARTICLE*.

Problémem tažení news je, že server potřebuje mít těsnou kontrolu nad skupinami a distribucemi, které povoluje klientovi požadovat. Musí se například ujistit, že nebude neautorizovaným klientům poslán tajný materiál z lokální diskusní skupiny.

Programy pro čtení news také disponují několika pohodlnými příkazy, které jim dovolují odděleně získání hlaviček a těl článků, případně i jednotlivých řádků článků. Takto je možné udržovat všechny news na centrálním hostiteli, k němuž mohou všichni uživatelé lokální sítě přistupovat prostřednictvím klientských programů pro čtení a posílání news, které pracující na bázi protokolu NNTP. Jedná se o alternativní řešení pro exportování adresářů news prostřednictvím systému NFS, což bylo popsáno v kapitole 17.

Celkový problém protokolu NNTP tkví v tom, že umožňuje informovanému člověku vložit do proudu news články s falešnou specifikací odesílatele. Tomu se říká *falšování news* (*news faking*)[3]. Rozšíření protokolu NNTP umožňuje u určitých příkazů vyžadovat autentifikaci uživatele.

V současné době existuje několik balíků protokolu NNTP. Jedním z nejpoužívanějších je démon NNTP, který je referenční implementací tohoto protokolu. Jeho původními autory jsou Stan Barber a Phil Lapsley, kteří chtěli jeho prostřednictvím ilustrovat detaily RFC 977. Nejaktuálnější verze má číslo `nntpd-1.5.11`, a tu si nyní popíšeme. Můžete si buď sehnat zdrojové soubory a sami si je přeložit, nebo použít program `nntpd` z balíku binárních souborů *net-std* od Freda van Kempena. Z důvodu různých konfiguračních nastavení, která jsou specifická pro jednotlivé systémy, nejsou poskytovány binární soubory balíku `nntpd`.

3 Stejný problém se vyskytuje v souvislosti s protokolem SMTP, Simple Mail Transfer Protocol.

Balík `nntpd` tvoří server a dva klienti pro tlačení a tažení news a najdete zde také náhradu programu `inews`. Vše je určeno pro prostředí Bnews, ale po několika úpravách bude fungovat i v prostředí C News. Pokud však plánujete používat protokol NNTP i k jiným účelům, než pro zprostředkování přístupu k vašemu serveru news, není pro vás probíraná implementace tou pravou volbou. Proto budeme probírat pouze démona NNTP z balíku `nntpd` a vynecháme popis klientských programů.

K dispozici je také balík nazvaný „InterNet News“, zkráceně INN, jehož autorem je Rich Salz. Umožňuje přenos jak NNTP, tak i UUCP-news a je vhodnější pro velké systémy news. Pro přenos news prostřednictvím protokolu NNTP je rozhodně lepší než `nntpd`. Balík INN je v současné době ve verzi `inn-1.4sec`. Existuje také sada nástrojů pro sestavení INN na linuxovém počítači. Jejím autorem je Arjan de Vet a je dostupná na serveru **sunsite.unc.edu** v adresáři `system/Mail`. Při konfiguraci balíku INN se řiďte dokumentací, která je dodávána společně se zdrojem, případně INN FAQ, které jsou pravidelně posílány do konference **news.software.b**.

18.2 Instalace serveru NNTP

Server NNTP se nazývá `nntpd` a lze ho zkompilovat dvěma způsoby, v závislosti na očekávaném zatížení systému news. Kvůli implicitním hodnotám specifickým pro některá místa, jež jsou zakódovány do spustitelného souboru, nejsou k dispozici žádné zkompilované verze. Veškerou konfiguraci provádí makro, které je definováno v souboru `common/conf.h`.

Server `nntpd` může být nakonfigurován buď jako samostatný server, který je spouštěn při zavádění systému z `rc.inet2`, nebo jako démon řízený `inetd`. Ve druhém případě je třeba mít v souboru `/etc/inetd.conf` následující záznam:

```
nntp   stream  tcp nowait   news   /usr/etc/in.nntpd   nntpd
```

Pokud konfigurujete `nntpd` jako samostatný server, ujistěte se, že je každý podobný řádek řádně okomentován. V každém případě se musíte přesvědčit, že v souboru `/etc/services` nechybí následující řádek:

```
nntp   119/tcp   readnews untp   # Network News Transfer Protocol
```

Aby bylo možné ukládat příchozí články apod., potřebuje mít server `nntpd` v adresáři `spool` také adresář `.tmp`. Vytvoříte ho následovně:

```
# mkdir       /var/spool/news/.tmp
# chown news.news /var/spool/news/.tmp
```

18.3 Omezení přístupu k NNTP

Přístup ke zdrojům NNTP je řízen souborem `nntp_access`, který je uložen v adresáři `/usr/lib/news`. Řádky v tomto souboru specifikují přístupová práva přidělená vzdáleným hostitelům. Každý řádek má následující formát:

site `read|xfer|both|no post|no [!exceptgroup]`

Jakmile se klient připojí na NNTP-port, pokusí se server `nntpd` pomocí zpětného výhledávání podle IP-adresy získat plně kvalifikovaný název domény. Hostitelský název klienta a jeho IP-adresa jsou porovnány s polem `site` každého záznamu v takovém pořadí, v jakém jsou uvedeny v souboru. Shoda může být buď částečná, nebo úplná. V případě úplné shody ji použije. V případě jen částečné shody ji použije jen pokud se nenajde žádná další částečná shoda. Pole `site` je možné specifikovat jedním z následujících způsobů:

hostname Toto je plně kvalifikovaný název domény hostitele. Pokud přesně odpovídá kanonickému hostitelskému názvu klienta, použije se tento záznam a všechny další záznamy budou ignorovány.

IP address Toto je IP-adresa ve tečkové notaci. Pokud jí IP-adresa klienta odpovídá, použije se tento záznam a všechny následující záznamy budou ignorovány.

domain name Toto je název domény zadaný ve formě *.domain. Pokud hostitelský název klienta odpovídá názvu domény, pak záznam vyhovuje.

network name Toto je název sítě specifikovaný v souboru `/etc/networks`. Pokud číslo sítě klientovy IP-adresy odpovídá číslu sítě sdružené s názvem sítě, pak záznam vyhovuje.

default Této volbě vyhovuje každý klient.

Záznamy s obecnější specifikací adres je dobré uvést na začátku souboru, protože případné shody budou pozdějšími přesnějšími shodami potlačeny.

Druhé a třetí pole popisují přístupová práva přidělená danému klientovi. Druhé pole uvádí přístupová práva pro získávání news metodou pull (*read*) a předávání news metodou push (*xfer*). Hodnota *both* povoluje oba přístupy, zatímco hodnota *no* přístup úplně zakazuje. Třetí pole uděluje klientovi právo předávat články, což znamená doručovat články s neúplnými hlavičkovými informacemi, které pak doplní software news. Pokud druhé pole obsahuje hodnotu *no*, je třetí pole ignorováno.

Čtvrté pole je volitelné a obsahuje čárkami oddělený seznam skupin, k nimž má klient zakázaný přístup.

Následuje ukázkový soubor `nntp_access`:

```
#
# všichni budou přebírat news, ale ne číst ani posílat
default                    xfer         no
#
# hostitel public.vbrew.com poskytuje veřejný přístup po modemu
# povolíme čtení a posílání do všech skupin kromě lokálních
public.vbrew.com           read         post   !local
#
# ostatní počítače pivovaru mohou číst i posílat
*.vbrew.com                read         post
```

18.4 Autorizace NNTP

Při realizaci přístupových symbolů typu *xfer* nebo *read* v souboru `nntp_access` vyžaduje server `nntpd` po klientovi autorizaci příslušných operací. Když například specifikujete přístupové právo *Xfer* nebo *XFER*, nepovolí server `nntpd` klientovi přenos článku do vašeho systému, pokud neprojde autentifikací.

Proces autentifikace je implementován prostřednictvím nového příkazu protokolu NNTP, který se nazývá *AUTHINFO*. Za pomoci tohoto příkazu předá klient serveru NNTP uživatelské jméno a heslo. Server `nntpd` je porovná s databází `/etc/passwd` a ověří, že příslušný uživatel patří do skupiny **nntp**.

18.5 Interakce serveru nntpd a systému C News

Po přijetí článku ho musí server `nntpd` doručit do subsystému news. V závislosti na tom, zda jej obdržel jako výsledek příkazu *IHAVE* nebo *POST*, je článek předán systému `rnews` nebo `inews`. Místo spouštění programu `rnews` jej také můžete (v době překladu) nastavit tak, aby zpracovával příchozí články do dávky a výsledné dávky pak posílal do souboru `/var/spool/news/in.coming`, kde si je při příštím dotazu vyzvedne program `relaynews`.

Aby mohl řádně používat protokol ihave/sendme, musí mít server `nntpd` přístup k souboru `history`. V době překladu se proto musíte ujistit, že máte správně nastavenu příslušnou cestu. Také je třeba se přesvědčit, že se systém C News a `nntpd` dohodli na formátu vašeho souboru history. Systém C News používá při přístupu k němu hašovací funkce `dbm`, která má

poměrně hodně různých a mírně odlišných implementací. Pokud by byl systém C News slinkován s jinou knihovnou `dbm`, než kterou máte ve vaší knihovně `libc`, musíte s touto knihovnou slinkovat také server `nntpd`.

Typickým symptomem nesourodosti databázového formátu serveru `nntpd` a systému C News jsou chybové zprávy v systémovém log-souboru, že server `nntpd` nemohl tuto databázi otevřít nebo prostřednictvím protokolu NNTP obdržel duplicitní články. Vhodným testem je vzít článek z adresáře `spool`, navázat telnetové spojení se serverem `nntpd` a nabídnout ho serveru `nntpd`, jak je demonstrováno v níže uvedeném příkladu (váš vstup je označen *takto*). Samozřejmě, že musíte nahradit řetězec <msg@id> řetězcem id zprávy článku, který chcete předat serveru `nntpd`.

```
$ telnet localhost nntp
Trying 127.0.0.1...
Connected to localhost
Escape characters is '^]'.
201 vstout NNTP[auth] server version 1.5.11t (16 November
1991) ready at Sun Feb 6 16:02:32 1194 (no posting)
IHAVE <msg@id>
435 Got it.
QUIT
```

Tento rozhovor ukazuje příslušnou reakci serveru `nntpd`; zpráva „Got it“ říká, že již příslušný článek má. Pokud místo toho obdržíte zprávu „335 Ok“, pak vyhledávání v souboru history z nějakého důvodu selhalo. Ukončete rozhovor stiskem kombinace kláves Ctrl+D. Co se stalo, zjistíte na základě systémového log-souboru; server `nntpd` zapisuje všechny druhy zpráv do souboru `syslog`. Nekompatibilní knihovnu `dbm` zpravidla zjistíte podle zprávy se stížností na selhání programu `dbminit`.

Kapitola

19

Konfigurace programu pro čtení news (Newsreader)

Cílem programů pro čtení news je zprostředkovat uživatelům snadný a komfortní přístup k funkcím systému news, jako je předávání článků nebo sbírání obsahu diskusní skupiny. Kvalita tohoto rozhraní je předmětem vášnivých debat.

Na platformu Linuxu bylo portováno několik programů pro čtení news. Dále si popíšeme základní nastavení tří nejoblíbenějších z nich, jmenovitě to budou `tin`, `trn` a `nn`.

Jedním z nejefektivnějších nástrojů pro čtení news získáte následujícím zápisem:

```
$ find /var/spool/news -name '[0-9]*' -exec cat {} \; | more
```

Tímto způsobem čtou news unixoví nadšenci.

Většina programů pro čtení news je však mnohem kultivovanější. Obvykle nabízí celoobrazovkové rozhraní s oddělenými úrovněmi nabízejícími zobrazení všech skupin, do kterých je uživatel přihlášen, zobrazení přehledu všech článků v jedné skupině a konečně i jednotlivých článků.

Na úrovni diskusní skupiny zobrazuje většina programů seznam článků, ve kterém je uveden předmět článku a jeho autor. Ve velkých skupinách nemůže uživatel sledovat vzájemnou provázanost článků, i když je možné vystopovat odpovědi na dřívější články.

V odpovědi je většinou zopakován předmět původního článku, před nějž je předsunut řetězec „`Re`:“. Kromě toho je id zprávy článku vytvořeno tak, aby přímo následovalo po předchozím a mohlo být umístěno do hlavičky `References`:. Seřazením článků podle těchto dvou kri-

terií vzniknou malé shluky (ve skutečnosti se jedná o stromy) článků, kterým se říká *vlákna* (*threads*). Jedním z úkolů při psaní programu pro čtení news je vymyslet efektivní schéma pro tvorbu vláken, protože čas potřebný pro takovéto seřazení je úměrný druhé mocnině celkového počtu článků

V této kapitole se však nebudeme hlouběji zabývat způsobem, jakým jsou vystavěna uživatelská rozhraní. Všechny programy pro čtení news, které jsou v současné době v Linuxu dostupné, disponují dobrou nápovědou, takže by vám práce s nimi neměla činit potíže.

Dále se budeme zabývat pouze administrativními úkoly. Většinou bude řeč o tvorbě databází vláken a evidencí.

19.1 Konfigurace programu tin

Nejvšestrannějším programem pro čtení news s ohledem na tvorbu vláken je program `tin`. Napsal ho Iain Leas a je volně odvozen od staršího programu jménem `tass`[1]. Propočet vláken provádí v době, kdy uživatel vstoupí do diskusní skupiny, a kromě případu, kdy používáte spojení NNTP, je poměrně rychlý.

Na počítači 486DX50 zabere setřídění 1000 článků při čtení přímo z disku zhruba 30 sekund. Při spojení NNTP se zatíženým serverem news by tato operace zabrala více než 5 minut.[2] Tuto dobu je možné zkrátit pravidelnou aktualizací indexového souboru, které dosáhnete pomocí volby `-u` nebo spuštěním programu `tin` s parametrem `-U`.

Program `tin` obvykle ukládá svou databázi vláken do souboru `.tin/index` v domovském adresáři uživatele. Tato konfigurace však může příliš zatěžovat systémové zdroje, takže možná budete chtít udržovat jedinou kopii těchto souborů na nějakém centrálním místě. Toho dosáhnete tak, že programu `tin` přidělíte například účet **news** nebo nějaký jiný účet s omezenými právy.[3] Program `tin` pak bude udržovat všechny databáze v souboru `/var/spool/news/.index`. Při každém přístupu k souboru nebo ukončení příkazového interpretu nastaví efektivní uid na skutečné uid uživatele, který ho spustil.[4]

1 Jeho autorem je Rich Skrenta.

2 K výraznému zlepšení dojde, pokud NNTP-server provede setřídění sám a klientovi předá přímo databáze vláken; umí to například program INN verze 1.4.

3 Pro tento účel však nepoužívejte účet **nobody**. Mělo by se dodržovat nepsané pravidlo, a totiž nesdružovat s tímto účtem žádné soubory nebo příkazy.

4 Z toho důvodu obdržíte nepěkné chybové zprávy, když tento program spustíte jako superuživatel. Kvůli tomu byste s ním neměli pracovat jako **root**.

Lepším řešením je nainstalovat indexovacího démona `tind`, který bude pravidelně aktualizovat indexové soubory. Tento démon však není součástí žádné distribuce Linuxu, takže si ho budete muset i sami přeložit. Pokud provozujete lokální síť s centrálním serverem news, můžete dokonce spouštět démona `tind` na tomto serveru a nechat klienty, aby si stahovali indexové soubory pomocí protokolu NNTP. K tomu samozřejmě potřebujete rozšíření protokolu NNTP. Doplňky démona `nntpd`, které zavádějí toto rozšíření, jsou součástí zdrojového kódu programu `tin`.

Verze programu `tin`, která byla součástí některých distribucí Linuxu, v sobě neměla zakomponovánu podporu protokolu NNTP, ale v současné době ji většina distribucí již má. Pokud program spustíte jako `rtin` nebo s volbou `-r`, pokusí se *tin* spojit s NNTP serverem uvedeným v souboru `/etc/nntpserver` nebo v proměnné prostředí `NNTPSERVER`. Soubor `nntpserver` obsahuje na jediném řádku pouze název serveru.

19.2 Konfigurace programu trn

Program `trn` je také následníkem staršího programu pro čtení news, jmenovitě je to program `rn` (což znamená *read news*). Písmeno „t“ v jeho názvu znamená „threaded“. Napsal ho Wayne Davidson.

Na rozdíl od programu `tin` neumí program `trn` generovat svou databázi vláken za běhu. Místo toho využívá databázi vytvořenou programem `mthreads`, který je pravidelně spouštěn z `cron` kvůli aktualizaci indexových souborů.

Když nespustíte program `mthreads`, neznamená to, že byste nemohli přistupovat k novým článkům, budete je ale všechny mít roztroušeny po výběrovém menu článků, místo jednoho vlákna, které by šlo lehce přeskočit.

Budete-li chtít zapnout tvorbu vláken pro konkrétní diskusní skupinu, spustíte program `mthreads` a jako argumenty mu na příkazové řádce předáte seznam diskusních skupin. Seznam vytvoříte stejným způsobem, jaký jste použili při tvorbě souboru `sys`:

```
mthreads comp,rec,!rec.games.go
```

Výše uvedený zápis povolí vytváření vláken pro všechny skupiny **comp** a **rec** s výjimkou skupiny **rec.games.go** (lidé, kteří hrají hru Go, nepotřebují líbivá vlákna). Potom spustíte stejný program bez argumentů, aby mohl zařadit nově příchozí články. Uspořádání všech skupin, které máte uvedeny v souboru `active`, lze zapnout pomocí argumentu **all**, který předáte programu `mthreads` v příkazové řádce.

Pokud dostáváte news přes noc, budete obvykle spouštět program `mthreads` jen jednou, a to ráno, ale dle potřeby to může být i častěji. Systémy s velkým provozem možná budou používat program `mthreads` v režimu démona. Pokud ho totiž spustíte při zavádění systému s parametrem `-d`, umístí sám sebe na pozadí a každých deset minut bude zjišťovat, zda nedošly nějaké nové články, a pokud ano, pak je zařadí. Chcete-li spouštět program `mthreads` jako démona, umístěte do skriptu `rc.news` následující řádek:

```
/usr/local/bin/rn/mthreads -deav
```

Parametr `-a` zapíná automatickou tvorbu vláken pro nové skupiny, parametr `-v` povoluje zápis zpráv do log-souboru programu `mthread`, který se nazývá `mt.log` a je umístěn v adresáři, ve kterém máte nainstalován program `trn`.

Staré články, které již nejsou k dispozici, je třeba z indexových souborů pravidelně mazat. Implicitně jsou odstraňovány pouze články, jejichž číslo je pod dolním vodoznakem (water mark).[5] Články nad tímto číslem, jejichž doba platnosti by přesto nevypršela (protože nejstaršímu článku mohla být za pomoci hlavičkového pole `Expires:` přidělena dlouhá doba platnosti), lze odstranit pomocí parametru `-e`, který spustí tzv. pokročilé vyhodnocování doby platnosti. Pokud program `mthreads` běží jako démon, způsobí parametr `-e`, že přejde do tohoto režimu jednou denně - krátce po půlnoci.

19.3 Konfigurace programu nn

Program `nn`, jehož autorem je Kim F. Storm, o sobě tvrdí, že je nástrojem, jehož cílem je nejenom umožňovat čtení news. Jeho název je zkratkou „No News“ a má následující moto: „No News is good news. `nn` is better.“ („Žádné zprávy – dobré zprávy. Program `nn` je lepší.“)

Aby dosáhl tohoto nesmělého cíle, přichází program `nn` s velkou zásobou podpůrných nástrojů, které dovolují nejenom generovat vlákna, ale umožňují také intenzívní kontrolu konzistence těchto databází a účtů, získávání statistiky využití a omezení přístupů. Součástí balíku je i administrativní program nazvaný `nnadmin`, který dovoluje provádět tyto úkoly interaktivně. Je velmi intuitivní, takže se jím zde nebudeme zabývat a zaměříme se pouze na vytváření indexových souborů.

Databáze vláken programu `nn` se nazývá `nnmaster`. Obvykle běží jako démon spuštěný ze skriptu `rc.news` nebo `rc.inet2`. Spouští se následovně:

```
/usr/local/lib/nn/nnmaster -l -r -C
```

5 Všimněte si, že systém C News neaktualizuje tento dolní vodoznak automaticky, ale je třeba spustit program `updatemin`, který to provede za něj. Podrobnosti viz kapitola 17.

Tento zápis povolí vytváření vláken pro všechny diskusní skupiny, které jsou uvedeny v souboru *active*.

Jinou možností je pravidelně spouštět program `nnmaster` z `cron` a přitom mu předat seznam příslušných skupin. Tento seznam je podobný seznamu zapsaných skupin v souboru `sys`. Liší se jen v tom, že místo čárek používá jako oddělovače mezery. Místo falešného názvu skupiny **all** byste měli k označení všech skupin používat prázdný řetězec „“. Program lze spustit například takto:

```
# /usr/local/lib/nn/nnmaster !rec.games.go rec comp
```

Všimněte si, že v tomto zápisu záleží na pořadí prvků v seznamu. Pokud bychom uvedli řetězec `!rec.games.go` až za řetězcem `rec`, byly by vždy zpracovány všechny články z této skupiny.

Program `nn` nabízí několik metod pro odstraňování článků, jimž vypršela doba platnosti. První z nich provádí aktualizaci databáze tak, že projde adresáře všech diskusních skupin a zruší ty záznamy, k nimž již neexistují odpovídající záznamy. Tato operace je implicitní a lze ji aktivovat i pomocí parametru `-E`. Kromě případu, kdy ji provádíte prostřednictvím NNTP, je poměrně rychlá.

Druhá metoda se chová úplně stejně jako program `mthreads` při implicitním spuštění. Odstraní totiž pouze ty záznamy, které odkazují na články, jejichž číslo je nižší než dolní vodoznak v souboru `active`. Tuto metodu aktivujete pomocí parametru `-e`.

Konečně třetí strategie spočívá ve zrušení celé databáze a opětovném sesbírání všech článků. Aktivujete ji, když program `nnmaster` spustíte s parametrem `-E3`.

Seznam skupin, jejichž platnost již vypršela, se předává pomocí parametru `-F` stejným způsobem. Pokud však spouštíte program `nnmaster` jako démona, musíte ho před aktualizací databáze „zabít“ (pomocí parametru `-k`) a znovu ho spustit s původními parametry. Příslušný příkaz pro spuštění kontroly vypršení platnosti všech skupin podle první metody tedy vypadá takto:

```
# nnmaster -kF ""
# nnmaster -lrC
```

Chování programu `nn` je možné upravit prostřednictvím mnoha dalších parametrů. Chcete-li vědět, jak odstraňovat špatné články nebo provádět výtahy z článků, pak si přečtěte manuálové stránky programu `nnmaster`.

Program `nnmaster` spoléhá na soubor nazvaný *GROUPS*, který je umístěn v adresáři `/usr/local/lib/nn`. Pokud neexistuje, program si ho sám vytvoří. Pro každou diskusní skupinu obsahuje tento soubor řádek začínající názvem této skupiny, za nímž může volitelně

následovat časové označení a příznaky. Úpravou těchto příznaků je možné změnit zacházení programu s touto skupinou. Nelze však změnit pořadí, ve kterém se skupiny objevují.[6] Přípustné příznaky a jejich význam také najdete v manuálových stránkách programu `nnmaster`.

6 To proto, že pořadí skupin musí souhlasit s pořadím záznamů v (binárním) souboru `MASTER`.

A

Nulový tiskový kabel pro PLIP

K sestavení nulového tiskového kabelu pro spojení PLIP potřebujete dva 25pinové konektory (označení DB-25) a nějaký 11vodičový kabel. Tento kabel může být dlouhý maximálně 15 metrů.

Podíváte-li se pozorně na konektor, měli byste vedle každého pinu přečíst drobná čísla, přičemž jedničku má pin vlevo nahoře (držíte-li konektor širší stranou nahoru) a pin v levém dolním rohu má číslo 25. Nulový tiskový kabel vytvoříte vzájemným propojením následujících pinů obou konektorů:

```
D0          2   ---   15     ERROR
D1          3   ---   13     SLCT
D2          4   ---   12     PAPOUT
D3          5   ---   10     ACK
D4          6   ---   11     BUSY
GROUND     25   ---   25     GROUND
ERROR      15   ---    2     D0
SLCT       13   ---    3     D1
PAPOUT     12   ---    4     D2
ACK        10   ---    5     D3
BUSY       11   ---    6     D4
```

Zbylé piny zůstanou nezapojené. Jedná-li se o stíněný kabel, může být stínění na jednom konci připojeno ke kovovému pouzdru.

B

Vzorové konfigurační soubory programu smail

Tento dodatek popisuje vzorové konfigurační soubory pro koncový uzel UUCP v lokální síti. Vycházejí z ukázkových souborů, které jsou součástí distribuce programu `smail-3.1.28`. I když se zde pokusíme o stručný výklad způsobu, jakým tyto soubory fungují, doporučujeme vám přečíst si velmi hezky zpracovanou manuálovou stránku `smail`*(8)*, kde jsou tyto soubory popsány lépe. Jakmile porozumíte základní myšlence konfigurace programu `smail`, zjistíte, že stojí za to si ji přečíst.

První je soubor `routers`, který popisuje sadu směrovačů pro program `smail`. Když má program `smail` doručit zprávu na danou adresu, předá tuto adresu postupně všem směrovačům, dokud některému z nich nevyhovuje, což zde znamená, že příslušný směrovač našel cílového hostitele ve své databázi, ať už je to soubor `paths`, `/etc/hosts` nebo jiný směrovací mechanismus, který směrovač používá.

Záznamy v konfiguračních souborech programu `smail` začínají jedinečným názvem, který identifikuje příslušný směrovač, přenos nebo direktor. Následuje seznam atributů, které definují jeho chování. Tento seznam je tvořen globálními atributy, jako například použitým ovladačem a soukromými atributy, kterým rozumí pouze konkrétní ovladač. Jednotlivé atributy jsou odděleny čárkami a sady globálních a soukromých atributů jsou navzájem odděleny středníkem.

Aby vám byl tento rozdíl jasný, předpokládejme, že chcete udržovat dva oddělené soubory s aliasy; jeden bude obsahovat směrovací informace pro vaši doménu a druhý pak globální směrovací informace, generované pravděpodobně z map UUCP. Pomocí programu `smail` nyní můžete v souboru `routers` specifikovat dva směrovače, které budou oba používat ovladač `pathalias`. Tento ovladač bude vyhledávat názvy hostitelů v databázi aliasů. Očekává, že mu bude jako soukromý atribut předán název souboru:

```
#
# pathalias database for intra-domain routing
domain_paths:
   driver=pathalias,     # look up host in a paths file
   transport =uux;       # if matched, deliver over UUCP
   file=paths/domain,    # file is /usr/lib/smail/paths/domain
   proto=lsearch,        # file is unsorted (linear search)
   optional,             # ignore if the file does not exist
   required=vbrew.com,   # look up only *.vbrew.com hosts
#
# pathalias database for routing to hosts outside our domain
world_paths:
   driver=pathalias,     # look up host in a paths file
   transport =uux;       # if matched, deliver over UUCP

   file=paths/world,     # file is /usr/lib/smail/paths/world
   proto=bsearch,        # file is sorted with sort(1)
   optional,             # ignore if the file does not exist
   -required,            # no required domains
   domain=uucp,          # string ending ".uucp" before searching
```


Druhý globální atribut uvedený v obou záznamech směrovačů definuje přenos, který má být použit v případě, že směrovač vyhovuje dané adrese. V našem případě bude zpráva doručena přenosem `uux`. Přenosy jsou definovány v souboru `transports`, který bude popsán dále.

Způsob přenosu si můžete lépe přizpůsobit, když místo atributu `transport` specifikujete soubor metod. Soubory metod poskytují mapování cílových hostitelů na přenosy. Zde se jimi však nebudeme zabývat.

Následující soubor `routers` definuje směrovače pro lokální síť, která se dotazuje knihovny resolveru. Na internetovém hostiteli byste však měli používat směrovač, který umí zpracovávat MX-záznamy. V tom případě je nutné zrušit komentáře u alternativního směrovače *inet_bind*, který používá zabudovaný ovladač BIND programu `smail`.

V prostředí, které kombinuje protokoly UUCP a TCP/IP možná narazíte na problém, kdy v souboru `/etc/hosts` máte hostitele, s nimiž dochází k jen příležitostnému spojení pomocí protokolů SLIP nebo PPP. Většinou jim budete chtít i nadále posílat poštu prostřednictvím protokolu UUCP. Aby si ovladač *inet_hosts* těchto hostitelů nevšímal, musíte je umístit do souboru `paths/force`. Jedná se o další databázi aliasů, která je prohledána ještě předtím, než program `smail` předá dotaz resolveru.

```
# A sample /usr/lib/smail/routers file
#
# force - force UUCP delivery to certain hosts, even when they are
in out /etc/hosts:
force:
   driver=pathalias,          # look up host in a paths file
   transport=uux;             # if matched, deliver over UUCP

   file=paths/force,          # file is /usr/lib/smail/paths/force
   optional,                  # ignore if the file does not exist
   proto=lsearch,             # file is unsorted (linear search)
   -required,                 # no required domains
domain=uucp,                  # string ending ".uucp" before sear-
ching

# inet_addrs - match domain literals containing literal
#         IP addresses, such as in janet@[191.72.2.1]
inet_addrs:
   driver=gethostbyaddr,      # driver to match IP domain literals
   transport=smtp;            # deliver using SMTP over TCP/IP

   fail_if_error,             # fail if address is malformed
   check_for_local,           # deliver directly if host is ourself

# inet_hosts - match hostnames with gethostbyname (3a)
#        Comment this out if you wish to use the BIND version inste-
ad
inet_hosts:
   driver=gethostbyname,      # match hosts with the library function
   transport=smtp;            # use default SMTP

   - required,                # no required domains
   -domain                    # no defined domain suffixes
   -only_local_domain             # don't restrict to defined domains

# inet_hosts - alternate version using BIND to access DNS
#inet_hosts:
#  driver=bind,               # use buil/in BIND driver
#  transport=smtp;            # use TCP/IP SMTP for delivery
```

```
#  defnames,                  # use standard domain searching
#  defer_no_connect           # try again if the nameserver is down
#  local_mx_okay              #fail (don't pass through) an MX to the
local host
#

# pathalias database for intra-domain routing
domain_paths:
   driver=pathalias,          # look up host in a paths file
   transport=uux;             # if matched, deliver over UUCP

   file=paths/domain,         # file is /usr/lib/smail/paths/domain
proto=lsearch,                # file is unsorted (linear search)
   optional,                  # ignore if the file does not exist
   required=vbrew.com,        # look up only *.vbrew.com hosts
#
# pathalias database for routing to hosts outside our domain
world_paths:
   driver=pathalias,          # look up host in a paths file
   transport =uux;            # if matched, deliver over UUCP

   file=paths/world,          # file is /usr/lib/smail/paths/world
   proto=bsearch,             # file is sorted with sort(1)
   optional,                  # ignore if the file does not exist
   -required,                 # no required domains
   domain=uucp,               # string ending ".uucp" before sear-
ching

# smart_hosts - a partically specified smarthost director
#  If the smart_path attribute is not defined in
#  /usr/lib/smail/config, this router is ignored.
#  The transport attribute is overriden by the global
#  smart_transport varriable
smart_host:
   driver=smarthost,          # special-case driver
   transport=uux;             # by default deliver over UUCP

   -path,                     # use smart_path config file variable
```

Manipulace s poštou, určenou pro lokální adresu, je řízena souborem `directors`. Ten je vystavěn podobným způsobenm, jako soubor `routers`, přičemž každá položka definuje jednoho direktora. Direktoři nedoručují zprávy, starají se pouze o potřebná směrování, například přes aliasy, postoupení pošty apod.

Při doručování zprávy na lokální adresu, například uživateli **janet**, předá program `smail` toto uživatelské jméno postupně všem direktorům. Pokud některému z nich vyhovuje, pak buď specifikuje přenos, kterým má být daná zpráva doručena (například do schránky daného uživatele), nebo vygeneruje novou adresu (například po vyhodnocení aliasu).

Z důvodů bezpečnosti obvykle direktoři provádějí spoustu kontrol, zda soubory, které používají, nemohly být pozměněny. Adresy získané pochybným způsobem (například ze souboru `aliases`, do kterého může kdokoliv zapisovat), jsou považovány za nebezpečné. Některé přenosové ovladače takovéto adresy odmítnou, například přenos, který doručuje zprávy do souboru.

Program `smail` kromě toho také sdružuje uživatele s každou adresou. Každá operace zápisu nebo čtení je prováděna jménem daného uživatele. Při doručování do schránky uživatele **janet**, je adresa samozřejmě sdružena s uživatelem **janet**. S jinými adresami, získanými například ze souboru `aliases`, jsou sdružena jiná uživatelká jména, například **nobody**.

Podrobnosti najdete na manuálové stránce programu `smail`*(8)*.

```
# A sample /usr/lib/smail/directories file

# aliasinclude - expand ":include:filename" addresses produced
#       by alias files
aliasinclude:
   driver=aliasinclude,       # use this special-case driver
   nobody;                    # access file as nobody user if unsecu-
re

   copysecure;                # get permissions from alias director
   copyowners,                # get owners from alias director

# forwardinclude - expand ":include:filename" addrs produced
#  by forward files
forwardinclude:
   driver=forwardinclude,     # use this special-case driver
   nobody;                    # access file as nobody user if unsecure
```

```
   checkpath,                # check path accessibility
   copysecure;               # get permissions from alias director
   copyowners,               # get owners from alias director

# aliases - search for alias expansions stored in a database
aliases:
   driver=aliasfile,         # general-purpose aliasing director
   -nobody,                  # all addresses are associated
                             # with nobody by default anyway
   sender_okay,              # don't remove sender from expansions
   owner=owner-$user;        # problems go to an owner address

   file=/usr/lib/aliases,    # default: sendmail compatible
   modemask=002,             # should not be globally writable
   optional,                 # ignore if file does not exist
   proto=lsearch,            # unsorted ASCII file

# dotforward - expand .forward files in user home directories
dotforward:
   driver=forwardfile,       # general-purpose forwarding director
   owner=real-$user,         # problems go to the users's mailbox
   nobody,                   # use nobody user, if unsecure
   sender_okay;              # sender never removed from expansion

   file=~/.forward,          # .forward file in home directories
   checkowner,               # the user can own this file
   owners=root,              # or root can own the file
   modemask=002              # it should not be globally writable
   caution=0-10:uucp:daemon, # don't run things as root or daemons
   # be extra careful of remotely accessible home directories
   unsecure="~ftp:~uucp:~nuucp:/tmp:/usr/tmp",

#forwardto - expand a "Forward to " line at the top of
# user's mailbox file
forwardto:
   driver=forwardfile,
   owner=Postmaster,         # errors go to Postmaster
```

```
   nobody,                    # use nobody user, if unsecure
   sender_okay;               # don't remove sender from expansion

   file=/var/spool/mail/${lc:user},            # location of
user's mailbox
   forwardto,                 # enable "Forward to " check
   checkowner,                # the user can own this file
   owners=root,               # or root can own the file
   modemask=0002              # under System V, group mail can write
   caution=0-10:uucp:daemon,  # don't run things as root or daemons

# user - match users on the local host with delivery to their mail-
boxes
user:  driver=user;           # driver to match usernames

   transport=local,           # local transport goes to mailboxes

#real_user - match usernames when prefixed with the string "real-"
real_user:
       driver=user;           # driver to match usernames

   transport=local,           # local transport goes to mailboxes
   prefix="real-",            # for example, match real-root

# lists  expand mailing lists stored below /usr/lib/smail/lists
lists: driver=forwardfile,
   caution,                   # flag all addresses with caution
   nobody,                    # and then associate the nobody user
   sender_okay,               # do NOT remove the sender
   owner=owner-$user;         # the list owner

   # map the name of mailing list to lower case
   file=lists/${lc:user}
```

Po úspěšném nasměrování zprávy předá program `smail` tuto zprávu přenosu, který specifikoval směrovač nebo direktor, jemuž tato zpráva vyhovovala. Tyto přenosy jsou definovány v souboru `transports`. Znovu opakujeme, že přenos je definován jako sada globálních a soukromých voleb.

Nejdůležitější volbou definovanou v každém záznamu je ovladač, který přenos obsluhuje, například ovladač *pipe*, který spustí příkaz uvedený v atributu *cmd*. Kromě toho existuje i několik globálních atributů, které může přenos využít a které provádějí různé transformace hlavičky zprávy a volitelně i těla zprávy. Atribut *return_path* například způsobí, že přenos vloží do hlavičky zprávy pole *return_fpath*. Atribut *unix_hack* způsobí, že před každý výskyt slova `From` na začátku řádku bude vložen znak >.

```
# A sample /usr/lib/smail/transports file

# local - deliver mail to local users
local:  driver=appendfile, # append message to a file
         return_path,      # include a Return-Path: field
         from,             # supply a From_ envelope line
         unix_from_hack,   # insert > before From in body
         local;            # use local forms for delivery

         file=/var/spool/mail/${lc:user}, # location of mailbox fi-
les
         group=mail,# group to own file for System V
         mode=0660, # group mail can access
         suffix="\n",      # append an extra newline

# pipe - deliver mail to shell commands
pipe:   driver=pipe,       # pipe message to another program
         return_path,      # include a Return-Path: field
         from,             # supply a From_ envelope line
         unix_from_hack,   # insert > before From in body
         local;            # use local forms for delivery

         cmd="/bin/sh -c $user", # send address to the Bourne Shell
         parent_env,       # environment info from parent addr
         pipe_as_user,     # use user-id associated with address
         ignore_status,    # ignore a non-zero exit status
         ignore_write_errors,     # ignore write errors, i.e., broken
pipe
         umask=0022,       # umask for child process
         -log_output,      # do not log stdout/stderr
```

```
# file - deliver mail to files
file:   driver=appendfile,
        return_path,      # include a Return-Path: field
        from,             # supply a From_ envelope line
        unix_from_hack,   # insert > before From in body
        local;            # use local forms for delivery

        file=$user,       # file is taken from address
        append_as_user,   # use user-id associated with address
        expand_user,      # expand ~ and $ within address
        suffix="\n",      # append an extra newline
        mode=0600,        # set permissions to 600

# uux - deliver to the rmail program on a remote UUCP site
uux:    driver=pipe,
        uucp,             # use UUCP-style addressing forms
        from,             # supply a From_ envelope line
        max_addrs=5,      # at most 5 addresses per invocation
        max_chars=200;    # at most 200 chars of addresses

        cmd="/usr/bin/uux - -r -a$sender -g$grade $host!rmail
$(($user)$)",
        pipe_as_sender,   # have uucp logs contain caller
        log_output,       # save error output for bounce messages
#       defer_child_errors,     # retry if uux returns an error

# demand - deliver to a remote rmail program, polling immediately
demand: driver=pipe,
        uucp,             # use UUCP-style addressing forms
        from,             # supply a From_ envelope line
        max_addrs=5,      # at most 5 addresses per invocation
        max_chars=200;    # at most 200 chars of addresses

        cmd="/usr/bin/uux - -a$sender -g$grade $host!rmail
$(($user)$)",
        pipe_as_sender,   # have uucp logs contain caller
        log_output,       # save error output for bounce messages
#       defer_child_errors,     # retry if uux returns an error
```

```
# hbsmtp - half-baked BSMTP. The output files must
#        be processed regularly and sent out via UUCP.
hbsmtp: driver=appendfile,
         inet,              # use RFC 822-addressing
         hbsmtp,            # batched SMTP w/o HELO and QUIT
         -max_addrs, -max_chars;  # no limit on number of addresses

         file="/var/spool/smail/hbsmtp/$host",
         user=root,         # file is owned by root
         mode=0600,         # only read-/writable by root.

# smtp - deliver using SMTP over TCP/IP
smtp:    driver=tcpsmtp,
         inet,
         -max_addrs, -max_chars; # no limit on number of addresses

         short_timeout=5m, # timeout for short operations
         long_timeout=2h,  # timeout for longer SMTP operations
         service=smtp,     # connect to this service port
# For internet use: uncomment the below 4 lines
#        use_bind,         # resolve MX and multiple A records
#        defnames,         # use standard domain searching
#        defer_no_connect, # try again if the nameserver is down
#        -local_mx_okay,   # fail an MX to the local host
```

C

Slovníček

Moto: Mohli bychom použít více záznamů a méně třpytu.
Nebojte se přispět svými návrhy.

Zapamatovat si, co znamenají všechny zkratky a termíny v oblasti sítí, je pro jednoho člověka nesmírně složité. Zde uvádíme ty z nich, které se v knize často objevovaly, a připojujeme k nim krátký popis.

ACU Automatic Call Unit. Modem.[1]

ARP Address Resolution Protocol. Protokol, který slouží k mapování IP-adresy na ethernetovou adresu.

ARPA Advanced Resarch Project Agency, později DARPA. Zakladatel Internetu.

ARPANET Praotec dnešního Internetu. Experimentální síť vytvořená v americké společnosti Defense Advanced Research Project Agency (DARPA).

Assigned Numbers (Přidělená čísla) Pravidelně zveřejňovaná část dokumentu *RFC*, která obsahuje seznam veřejně přidělených čísel používaných pro různé účely v sítích TCP/IP. Obsahuje například seznam všech čísel portů známých služeb jako je *rlogin*, *telnet* atd. Nejaktuálnější vydání tohoto dokumentu má název RFC 1340.

---

1 Eventuelně teenager s telefonem.

bang path V sítích UUCP se jedná o zvláštní notaci cesty z jednoho systému UUCP na jiný. Název vznikl podle vykřičníků (‘bangs‘), které slouží k oddělení názvů hostitelů. Například cesta **foo!bar!ernie!bert** určuje cestu k hostiteli **bert**, při níž se půjde (v tomto pořadí) přes hostitele **foo**, **bar** a **ernie**.

BBS Bulletin Board System. Vytáčený systém pro předávání zpráv elektronickou poštou.

BGP Border Gateway Protocol. Protokol pro výměnu směrovacích informací mezi autonomními systémy.

BIND Berkeley Internet Name Domain server. Implementace serveru DNS.

BNU Basic Networking Utilities. V současné době nejběžnější varianta protokolu UUCP. Také se jí někdy říká HoneyDanBer UUCP. Tento název je odvozen ze jmen autorů: P. Honeyman, D. A. Novitz a B. E. Redman.

broadcast network (vysílací síť) Síť, která umožňuje jedné stanici současnou adresaci datagramu pro všechny ostatní stanice v síti.

BSD Berkeley Software Distribution. Druh Unixu.

canonical hostname (kanonický název hostitele) Primární název hostitele v DNS. Je to jediný název hostitele, se kterým je sdružen A-záznam a který je vrácen při provádění zpětného vyhledávání.

CCITT Comiteé Consultatif International de Télegraphique et Téléphonique. Mezinárodní organizace telefonních služeb.

CSLIP Compressed Serial Line IP. Protokol pro výměnu IP-paketů po sériové lince, přičemž dochází ke kompresi hlaviček většiny datagramů TCP/IP.

DNS Domain name system. Distribuovaná databáze, která se používá v Internetu pro mapování názvů hostitelů na IP-adresy.

EGP External Gateway Protocol. Protokol pro výměnu směrovacích informací mezi autonomními systémy.

Ethernet V hovorové mluvě se používá pro označení druhu síťového vybavení. Z technického hlediska je Ethernet jedním ze sady standardů IEEE. Ethernet jako hardware používá ke spojení několika hostitelů jediný

kabel, často koaxiál, který umožňuje přenosové rychlosti až 10 Mbp. Protokol Ethernet definuje způsob, jakým mohou hostitelé komunikovat prostřednictvím tohoto kabelu.[2]

FQDN Fully Qualified Domain Name. Název hostitele s připojeným názvem domény, který je platným indexem v databázi názvů domén.

FTP File Transfer Protocol. Protokol, na kterém je založena a podle něj i pojmenována jedna z nejznámějších služeb pro přenos souborů.

FYI „For Your Information" (Pro vaši informaci). Sada dokumentů obsahující neformální informace týkající se Internetu.

GMU Groucho March University. Fiktivní univerzita používaná v knize jako příklad.

GNU GNU není Unix – tato rekurzivní zkratka je názvem projektu nadace Free Software Foundation, jehož cílem je poskytovat kompletní sadu unixových nástrojů, které je možno volně používat a kopírovat. Na veškerý software GNU se vztahuje speciální poznámka o autorských právech, které se také říká GNU General Public License (GPL), neboli Copyleft. Tuto licenci najdete v dodatku D.

HoneyDanBer Název varianty protokolu UUCP. Viz též BNU.

host (hostitel) Obecně uzel sítě. Zařízení, které je schopno přijímat a vysílat síťové zprávy. Zpravidla se jedná o počítač, ale mohou jím být také X-terminály nebo chytré tiskárny.

ICMP Internet Control Message Protocol. Síťový protokol, který používá protokol IP, když vrací hostiteli-odesilateli chybové informace atd.

IEEE Institute of Electrical and Electronics Engineers. Další organizace zabývající se tvorbou standardů. Z pohledu unixového uživatele jsou jejím největším úspěchem pravděpodobně standardy POSIX, které definují vlastnosti unixových systémů od rozhraní a sémantiky systémových volání až po administrativní nástroje.

Kromě toho vyvinula firma IEEE specifikace pro sítě Ethernet, Token Ring a Token Bus. Dílem IEEE je také hojně používaný standard pro binární reprezentaci reálných čísel.

[2] Jen tak mimochodem, ethernetový protokol, který běžně využívá protokol TCP/IP, neodpovídá přesně standardu IEEE 802.3. Ethenetové rámce obsahují typové pole, zatímco rámce IEEE 802.3 obsahují délkové pole.

IETF Internet Engineering Task Force.

internet Počítačová síť tvořená několika menšími samostatnými sítěmi.

Internet Konkrétní celosvětový internet.

IP Internet Protocol. Síťový protokol.

ISO International Standards Organization. Společnost zabývající se prosazováním standardů.

ISDN Integrated Services Digital Network. Nová telekomunikační technologie, která používá místo analogového systému digitální.

LAN Local Area Network. Malá počítačová síť.

MX Mail Exchanger. Typ zdrojového záznamu DNS používaného k označení hostitele jako poštovní brány pro nějakou doménu.

network, packet-switched
(přenos paketů) Paleta sítí, která umožňuje okamžitý přenos dat po malých paketech, které jsou jednotlivě přeneseny na místo určení. Sítě typu packet-switched spoléhají na trvalé nebo skoro trvalé spojení.

network, store-and-forward
(ulož a pošli) Jedná se o protiklad sítí typu packet-switched. Tyto sítě přenášejí data po celých souborech a nepoužívají trvalá spojení. Místo toho se hostitelé navzájem spojují pouze v určitých intervalech a přenášejí všechna data najednou. Tento způsob vyžaduje dočasné uložení dat, než je spojení navázáno.

NFS Network File System. Standardní síťový protokol a softwarový balík zajišťující transparentní přístup k datům na vzdálených discích.

NIS Network Information System. Aplikace založená na protokolu RPC, která umožňuje sdílet konfigurační soubory (například soubor s hesly) mezi několika hostiteli. Viz též heslo YP.

NNTP Network News Transfer Protocol. Protokol používaný pro přenos news po síťových TCP-spojeních.

oktet V Internetu se jedná o technický termín, který označuje posloupnost osmi bitů. Používá se spíše než *bajt*, protože na některých počítačích v Internetu má bajt jiný počet bitů než osm.

OSI Open Systems Interconnection. Standard ISO pro síťový software.

path (cesta) Tento termín je v sítích UUCP často používán jako synonymum pro směr (*route*). Viz též *bang path*.

PLIP Parallel Line IP. Protokol pro výměnu IP-paketů po paralelní lince, například tiskový port.

port, TCP nebo UDP Porty jsou pro protokoly TCP a UDP abstrakcí koncového bodu služby. Dříve než může proces zprostředkovat přístup nebo přistupovat k nějakému síťovému zařízení, musí požádat o port. Společně s IP-adresou hostitele slouží porty k jedinečné identifikaci dvou účastníků TCP-spojení.

portmapper (mapovač portů) Jedná se o prostředníka mezi čísly programů, která používá RPC kvůli identifikaci jednotlivých RPC-serverů a čísel portů protokolů TCP a UDP, na kterých tyto služby odposlouchávají.

PPP Protokol point-to-point. Protokol PPP je flexibilní a rychlý podkladový protokol sloužící k posílání síťových protokolů typu IP nebo IPX po spojení typu point-to-point. Kromě využití v sériových (modemových) linkách lze tento protokol použít také jako podkladový protokol pro ISDN.

RARP Reverse Address Resolution Protocol. Povoluje hostiteli nalézt svou IP-adresu při zavádění systému.

resolver Tato knihovna je zodpovědná za mapování názvů hostitelů na IP-adresy a vice versa.

resource record (zdrojový záznam) Základní jednotka informace v databázi DNS, jejíž označení se často zkracuje na RR. Každý záznam má přidělen určitý typ a třídu, například záznam mapující název hostitele na IP-adresu má typ A (jako adresa) a třídu IN (jako Internet Protocol).

reverse lookup (zpětné vyhledávání) Proces vyhledávání názvu hostitele podle jeho IP-adresy. V rámci DNS se tak děje vyhledáním IP-adresy hostitele v doméně **in-addr.arpa**.

RFC Request For Comments. Sada dokumentů popisujících internetové standardy.

RIP Routing Information Protocol. Tento směrovací protokol se používá k dynamickému přizpůsobování tras uvnitř (malé) sítě.

route (trasa, směr) Posloupnost hostitelů, kterými musí informace projít při cestě od původního hostitele k cílovému. Hledání vhodné trasy se také říká *směrování*.

routing daemon (směrovací démon) Ve větších sítích se změny v topologii obtížně zavádějí, a proto se k šíření směrovacích informací členským hostitelům sítě používají jiné prostředky. Těmto prostředkům říkáme dynamické směrování, kdy si směrovací informace mezi sebou vyměňují *směrovací démoni*, kteří běží na centrálních hostitelích v síti. Protokoly, které k tomu využívají, se nazývají *směrovací protokoly*.

RPC Remote Procedure Call. Protokol pro provádění procedur uvnitř procesu na vzdáleném hostiteli.

RR Zkratka *resource record*.

RS-232 Běžný standard sériových rozhraní.

RTS/CTS Hovorový název hardwarového „podání ruky“, který provádí dvě zařízení komunikující spolu přes rozhraní RS-232. Název je odvozen ze dvou okruhů: RTS („Ready To Send“) a CTS („Clear To Send“).

RTM Internet Worm (Internetový červ) Virus, který se díky několika bezpečnostním dírám v Unixu VMS a BSD 4.3 rozšířil po Internetu. Několik „chyb“ v programu způsobilo jeho nekontrolovatelné množení a efektivní kolaps velkých částí Internetu. RTM jsou iniciály autora (Roberta T. Morrise), které nechal v těle programu.

site (systém, místo) Seskupení hostitelů, které se zvenku chová téměř jako jediný síťový uzel. Například z hlediska Internetu bychom mohli považovat Groucho Marx University za jeden systém bez ohledu na komplexnost jeho vnitřní sítě.

SLIP Serial Line IP. Protokol pro výměnu IP-paketů po sériové lince, viz též CSLIP.

SMTP Simple Mail Transfer Protocol. Používá se k poštovnímu přenosu po TCP-linkách, ale také pro přenos poštovních dávek po UUCP-linkách (dávkované SMTP).

SOA Start of Authority. Typ zdrojového záznamu DNS.

System V Druh Unixu.

TCP Transmission Control Protocol. Síťový protokol.

TCP/IP Nedbalý popis sady internetového protokolu jako celku.

UDP User Datagram Protocol. Síťový protokol.

UUCP Unix to Unix Copy. Sada síťových přenosových příkazů pro vytáčené sítě.

Version 2 UUCP Zastarávající varianta protokolu UUCP.

virtual beer (virtuální pivo) Oblíbený nápoj každého uživatele Linuxu. První zmínka o virtuálním pivu, pokud si pamatuji, byla v poznámkách k jádru Linuxu verze 0.98.X, kde se Linus zmiňoval o „Oxford Beer Trolls“ kvůli zaslání nějakého virtuálního piva.

well-known services (dobře známé služby) Tento termín se často používá pro běžné síťové služby, jako je *telnet* a *rlogin*. Z technického hlediska popisuje všechny služby, kterým bylo v dokumentu „Assigned Numbers“ RFC přiřazeno oficiální číslo portu.

YP Yellow Pages (Žluté stránky). Starší název služby systému NIS, který se již nepoužívá, protože se jedná o obchodní značku společnosti British Telecom. Nicméně většina utilit NIS si i nadále ponechala názvy s předponou *yp*.

D

Seznam literatury

Knihy

Z knih v následujícím seznamu se dozvíte více informací o některých tématech probíraných v této knize. Seznam není ani úplný, ani systematický, knihy zde uvedené jsem shodou okolností četl a připadaly mi poměrně užitečné. Jakékoliv doplňky a vylepšení tohoto seznamu jsou vítány.

Obecné knihy o Internetu

[Kehoe92] Brendan P. Kehoe: *Zen and the Art of the Internet.*

„Zen“ byl jedním z prvních, ne-li vůbec prvním intenetovým průvodcem, který seznamoval začínajícího uživatele s různými trendy, službami a folklórem kolem Internetu. Jakožto stostránkový svazek pokrýval témata od přispívání do usenetových news až po Internetového červa. Kniha je k dispozici prostřednictvím anonymní služby FTP na mnoha FTP serverech a lze ji volně šířit a publikovat. Tištěná verze je k dostání také v nakladatelství Prentice-Hall.

Administrativní problémy

[Hunt92] Craig Hunt: *TCP/IP Network Administration.* O'Reilly and Associates, 1992. ISBN 0-937175-82-X.

Není-li pro vás dost dobrý Network Administration Guide, sežeňte si tuto knihu. Najdete v ní vše od získání IP-adresy až po řešení síťových problémů a bezpečnostní aspekty.

Zaměřuje se na nastavení protokolu TCP/IP, tj. konfiguraci rozhraní, nastavení směrování a rozpoznávání názvů. Obsahuje podrobný popis vlastností, které nabízí směrovací démoni `routed` a `gated`, včetně dynamického směrování.

Najdete zde také popis konfigurace aplikačních programů a síťových démonů, jako je `inetd`, tzv. příkazy `r`, NIS a NFS.

Dodatek obsahuje podrobný popis démonů `gated` a `named` a popis konfigurace programu sendmail.

[Stern92] Hal Stern: *Managing NIS a NFS*. O'Reilly and Associates, 1992. ISBN 0-937175-75-7.

Jedná se o doprovodnou knihu ke knize Craiga Hunta „TCP/IP Network Administration". V plném rozsahu se zabývá použitím systémů NIS (Network Information System) a NFS (Network File System), včetně konfigurace automounteru a systému PC/NFS.

[OReilly89] Tim O'Reilly and Grace Todino: *Managing UUCP and Usenet, 10th ed.* O'Reilly and Associates, 1992. ISBN 0-93717593-5.

Toto je standardní kniha o sítích UUCP. Popisuje UUCP Version 2, stejně jako verzi BNU. Pomůže vám rozchodit UUCP-uzel od úplného začátku a najdete zde také praktické rady a řešení mnoha problémů, jako je testování spojení nebo psaní skriptů pro rozhovory. Kniha se zabývá také exotičtějšími tématy, například jak zřídit cestovní UUCP-uzel nebo jemnostmi různých odrůd protokolu UUCP.

Druhá část knihy se zabývá Usenetem a softwarem pro síťové news. Vysvětluje konfiguraci jak Bnews (verze 2.11), tak i C News a seznámí vás se základy údržby síťových news.

[Spaf93] Gene Spafford and Simson Garfinkel: *Practical UNIX Security*. O'Reilly and Associates, 1992. ISBN 0-937175-72-2.

Tato kniha je nezbytná pro každého správce systému s přístupem k síti a nejenom pro něj. Probírá všechny sporné body týkající se počítačové bezpečnosti, které jsou seřazeny od základních bezpečnostních funkcí, jež Unix nabízí. I když byste se měli snažit zabezpečit všechny části systému, je část věnovaná sítím a bezpečnosti z hlediska našeho kontextu nejzajímavější. Vedle základních bezpečnostních strategií, které se týkají služeb BSD (*telnet*, *rlogin* atd), systémů NFS a NIS, se kniha zabývá také pokročilými bezpečnostními rysy typu MIT Kerberos, Secure RPC od firmy Sun a použití firewallů k odstínění vaší sítě před útoky z Internetu.

[AlbitzLiu92] Paul Albitz and Cricket Liu: *DNS and BIND*. O'Reilly and Associates, 1992. ISBN 1-56592-010-4.

Tato kniha je užitečná pro ty, kteří se zabývají správou systému DNS. Podrobně vysvětluje všechny rysy služby DNS a poskytuje příklady, které vám zpříjemní dokonce i volby BIND, jež vypadají na první pohled zcela nepřirozeně. Skutečně se dobře čte a hodně jsem se toho z ní dozvěděl.

[NISPlus] Rick Ramsey: *All about Administering NIS+*. Prentice-Hall, 1993. ISBN 0-13-068800-2.

Pozadí

Následující seznam knih možná bude zajímat ty, kdo se chtějí dozvědět více o tom, jak funguje protokol TCP/IP a jeho implementace, ale nechtějí číst dokumenty RFC.

[Stevens90] Richard W. Stevens: *UNIX Network Programming*. Prentice-Hall International, 1990. ISBN 0-13-949876-X.

Toto je zřejmě nejpoužívanější kniha o programování v sítích TCP/IP, z níž se současně dozvíte vše možné o internetových protokolech.[1]

[Tanen89] Andrew S. Tanenbaum: *Computer Networks*. Prentice-Hall International, 1989. ISBN 0-13-166836-6.[2]

Tato kniha vám dá nahlédnout do obecných síťových problémů. Za pomoci referenčního modelu OSI vysvětluje problémy spojené s návrhem každé vrstvy a algoritmy, které k tomu slouží. Na úrovni každé vrstvy jsou navzájem srovnávány implementace několika sítí, mezi nimiž nechybí ani ARPANET. Jedinou nevýhodou této knihy je příliš velké množství zkratek, takže je někdy obtížné sledovat, co autor říká. Tento problém je ale síťovému prostředí vrozený.

[Comer88] Douglas R.Comer: *Internetworking with TCP/IP: Principles, Protocols, and Architecture*. Prentice-Hall International, 1988.

Dokumenty RFC

Na následující seznam dokumentů RFC se odkazujeme v průběhu celé knihy. Všechny dokumenty RFC jsou k dispozici prostřednictvím anonymní služby FTP na adrese **nic.ddn.mil, ftp.uu.net**. Chcete-li dostat RFC elektronickou poštou, pošlete zprávu na adresu **service@nic.ddn.mil** a jako předmět této zprávy uveďte `RFC-number.TXT`.

1340 Assigned Numbers, *Postel, J.,* and *Reynolds, J.* Tento dokument RFC definuje významná čísla používaná v různých protokolech, například standardní čísla portů, které odposlouchávají servery TCP a UDP, a čísla protokolů používaná v hlavičce IP-datagramu.

1 Nezapomeňte, že Stevens napsal novou knihu o protokolu TCP/IP nazvanou *TCP/Illustrated, Volume 1, The Protocols*, která vyšla v nakladatelství Addison Wesley. Zatím jsem neměl čas se na ni podívat.

2 ISBN, pod kterým je k dostání v Severní Americe, se může lišit.

1144 Compressing TCP/IP headers for low-speed seriál links, *Jacobson, V.* Tento dokument popisuje algoritmus používaný při kompresi hlaviček TCP/IP v protokolech CSLIP a PPP. Velice hodnotné čtení.

1033 Domain Administrators Operations Guide, *Lottor, M*. Společně s doprovodnými dokumenty, RFC 1034 a RFC 1035, tvoří ucelený zdroj informací o systému DNS, Domain Name System.

1034 Domain Names – Concepts and Facilites, *Mockapetris, P.V*. Doprovodný dokument k dokumentu RFC 1033.

1035 Domain Names – Implementation and Specification, *Mockapetris, P.V*. Doprovodný dokument k dokumentu RFC 1033.

974 Mail Routing and the Domain System, *Partridge, C*. Tento dokument popisuje směrování pošty na Internetu. Zájemci zde najdou úplný popis MX-záznamů.

977 Network News Transfer Protocol, *Kantor, B*., and *Lapsley, P*. Definice NNTP, všeobecně používaného protokolu pro přenos news v Internetu.

1094 NFS: Network File System Protocol Specification, *Nowicki, B*. Formální specifikace protokolu NFS a připojovacích protokolů (verze 2).

1055 Nonstandard for Transmission of IP Datagrams Over Seriál Lines: SLIP, *Romkey, J.L.* Popisuje protokol SLIP, Seriál Line Internet Protocol.

1057 RPC: Remote Procedure Call Protocol Specification: Version 2, *Sun Microsystems, Inc*.

1058 Routing Information Protocol, *Hedrick, C.L.* Popisuje protokol RIP, který se používá při dynamické výměně směrovacích informací mezi sítěmi LAN a MAN.

821 Simple Mail Transfer Protocol, *Postel, J.B.* Definuje SMTP, poštovní přenosový protokol používaný v sítích TCP/IP.

1036 Standard for the Interchange of USENET messages, *Adams, R.,* and *Horton, M.R.*. Tento dokument RFC popisuje formát zpráv usenetových news a způsob jejich výměny v Internetu a v sítích UUCP. V brzké době bychom se měli dočkat revize tohoto dokumentu.

822 Standard for the Format of ARPA Internet text messages, *Crocker, D*. Úplný zdroj vědomostí co se týče pošty přizpůsobené standardu RFC. Každý ho zná, ale jen málo lidí ho skutečně četlo.

968 Twas the Night Before Start-up, *Cerf, V.* Kdo říká, že hrdinové sítí nejsou opěvováni?