UNIX

Programování a správa systému II

Jan Kasprzak (kas@fi.muni.cz)

UNIX je user-friendly,
ale své přátele si vybírá.

Úvodní informace

Předpoklady

- **Programování v C a v POSIX.1 rozhraní** (na úrovni PV065).
- **UNIX z uživatelského hlediska**.
- **Síť TCP/IP z uživatelského hlediska**.

Cíle kursu

- **Základy správy systému**
- **Znalost architektury TCP/IP**
- **Základy programování síťových aplikací**
- **Základy konfigurace síťových protokolů/služeb**

Ukončení předmětu

- **Test** – cca 20 otázek.
- **Hodnocení**: –1 až 2 body na otázku, je potřeba 20 bodů a více.

Obsah přednášky

- **Základy administrace systému** – instalace, údržba, zálohování.
- **Systém souborů a adresářů** – rozložení v systému.
- **Start systému** – `init`, inicializační scripty.
- **Uživatelé a skupiny** – data, PAM, nsswitch, programování.
- **Subsystémy** – `syslog`, `cron`, tiskárny, diskové kvóty.

- **Základy architektury sítě TCP/IP** – vrstvy sítě, formáty packetů.
- **Programování síťových aplikací** – BSD sockets API.

- **Konfigurace sítě** – ARP, interface, směrování.
- **Sériová komunikace** – modemy, SLIP, PPP, proxy-ARP.
- **DNS a překlad adres** – BIND, architektura, konfigurace.
- **Konfigurace služeb sítě** – `inetd`, TCP wrapper, FTP.
- **Vzdálené přihlášení** – protokoly telnet, rlogin, secure shell.
- **Vzdálené volání procedury** – RPC a XDR, portmapper, NFS, NIS/YP.
- **Uživatelské informace po síti** – LDAP, Kerberos.
- **Elektronická pošta** – formát zpráv, SMTP, POP-3, IMAP.
- **Firewally** – packetové filtry, aplikační brány.
- **X Window System** – architektura, programování a správa.
- **IPv6** – základní informace.

Materiály ke studiu

- **Slidy z přednášek:** `http://www.fi.muni.cz/~kas/p077/`

- **Pavel Šmrha & Vladimír Rudolf:** *Internetworking pomocí TCP/IP*, KOPP 1994, ISBN 80–85828–09–X
- **W. Richard Stevens:** *UNIX Network Programming*, Prentice-Hall 1990
- **Olaf Kirch:** *Linux Network Administrator's Guide*, O'Reilly & Associates, 1995, ISBN 1–56592–087–2
- **Craig Hunt:** *TCP/IP Network Administration*, O'Reilly & Associates, 1992, ISBN 0–937175–82–X
- **Paul Albitz & Cricket Liu:** *DNS and BIND*, O'Reilly & Associates, druhé vydání 1996, ISBN 1–56592–236–0
- **Bryan Costales & Eric Allman:** *Sendmail*, O'Reilly & Associates, druhé vydání 1997, ISBN 1–56592–222–0
- **Simson Garfinkel & Gene Spafford:** *Practical UNIX & Internet Security*, O'Reilly & Associates, druhé vydání 1996, ISBN 1–56592–148–8
- **Pavel Satrapa:** *Linux – Internet Server*, Neokortex 1998

Instalace systému

- **Start jádra z instalačního média.**
- **Rozdělení disků na oblasti** – `fdisk(8)`, `divvy(8)`, `disklabel(8)`.
- **Vytvoření souborových systémů** – `mkfs(8)`, `newfs(8)`, `mke2fs(8)`.
- **Inicializace odkládacího prostoru** – `mkswap(8)`, není nutná na všech systémech.
- **Minimální systém** – připojení z CD, z NFS nebo instalace do swapovací oblasti.
- **Počáteční konfigurace hardware** – vytvoření jádra pro nový systém.
- **Instalace jednotlivých částí systému.**
- **Post-instalační konfigurace systému** – přidělení doménového jména počítače, konfigurace sítě, časové zóny, systémového hesla a podobně.
- **Restart nainstalovaného systému.**

Po instalaci

- **Nastavit čas poslední modifikace** všech souborů na nějakou definovanou hodnotu:

```
# find / \! -type l \! -type d -print0 \
       | xargs -0 touch -t 010100001990 --
```

(systémy balíků typu RPM nebo DEB dělají za vás).

- **Vytvořit kontrolní součty souborů** kopii kontrolních součtů držet i na nějakém zařízení mimo počítač:

```
# find / -type f -print0 \
       | xargs -0 md5sum -- >/etc/sums
```

(lze použít například `tripwire`).

- **Zazálohovat** nově nainstalovaný systém na vnější zařízení.

Zálohování

Proč zálohovat:

- Ochrana dat před nechtěným smazáním.
- Ochrana dat před výpadkem hardwaru.
- Možnost sledování změn v datech.
- Možnost obnovení dat po bezpečnostním incidentu.

Problémy:

- Nízká rychlost zálohovacích médií – nejde o snímek systému.
- Malá velikost médií – nelze každý den zálohovat všechno.
- Nespolehlivost médií – je nutno mít několik *sad* záloh.

Víceúrovňové zálohování

- Řeší problém rychlosti a velikosti zálohovacího média.
- Záloha úrovně 0 – kompletní svazek nebo adresář.
- Záloha úrovně $n + 1$ – soubory a adresáře, modifikované od *začátku* zálohy úrovně n.
- Zálohovací systém by měl rozpoznat i smazané soubory.
- Čas vytvoření zálohy dané úrovně musí být uložen *na zálohovacím médiu* kde je tato záloha uložena, nikoli na disku.

Formát záloh

- **Vlastnický formát** – nevýhoda – je nutno mít kdykoli (a nejlépe i kdekoli) možnost zálohu rozbalit.
- **tar** – neumožňuje zabalit jen některé soubory (GNU tar ano)
- **cpio** – pozor na zabalení s absolutní cestou.
- **dump** – formát příslušný určitému typu souborového systému. Odpovídající `restore(8)` obvykle umí běžet nad libovolným FS.
- **Centrální zálohovací systémy** – páskové knihovny. Např. Legato Networker.

- **On-line záloha** – replikace databáze, kopírování na jiný počítač, zrcadlení přes `nbd`.
- **Off-site backupy** – proti živelným pohromám a krádeži.
- **Zabezpečení** – šifrovaný nebo zamčený backup.
- **Uchovávat i hodně staré backupy.**

Rozložení adresářů v systému

- **Tradiční** – není specifikováno žádnou de iure normou.
- **Rozdíly** – BSD versus System V, modifikace od jednotlivých výrobců.
- **Linux** – FileSystem Hierarchy standard (FHS).

Kořenový adresář

- Obvykle malý svazek, který není sdílen mezi více stroji.
- Programy by neměly vyžadovat vytváření dalších souborů nebo adresářů přímo pod `/`.

/bin – nezbytné uživatelské programy (přístupné všem uživatelům). Měly by zde být pouze programy, nutné k jednouživatelskému běhu systému a k nastartování sítě).

/boot – statické soubry zavaděče systému (tento adresář by měl být na svazku, který je dostupný firmwaru).

/dev – speciální soubory. Často také obsahuje program `MAKEDEV` pro vytváření těchto speciálních souborů.

/etc – konfigurační soubory pro konkrétní systém. Nelze sdílet mezi počítači. Na systémech blízkých SysV zde jsou také programy, které používá správce systému (viz `/sbin`).

/home – domovské adresáře uživatelů. Mohou být i jinde, záleží na správci. Obvykle samostatný svazek.

/lib – sdílené knihovny, nezbytné pro běh systému. Plug-iny a další data.

/opt – přidané softwarové balíky. Tento adresář bývá často na zvláštním svazku.

/root – domovský adresář superuživatele. Někdy též `/`.

/sbin – systémové programy (programy, které používá jen systém sám nebo správce systému). Na některých systémech chybí a tyto programy jsou v `/etc`.

/tmp – dočasné soubory. Adresář, přístupný všem užívatelům (vyžaduje sticky bit). Často se doporučuje, aby tento adresář byl promazán při startu systému.

Adresář `/usr`

- Sdílitelná data, přístupná v běžném případě pouze pro čtení.
- Subsystémy by neměly vytvářet další adresáře pod `/usr` – jedinou výjimkou je X11 z tradičních důvodů.

X11 – X Window System (často též `X11R6`). Obsahuje mj. i podadresáře `bin`, `lib` a `include` s odkazy z adresářů `/usr/bin`, `/usr/lib` a `/usr/include`.

bin – uživatelské programy, které nejsou nezbytné v jednouživatelském režimu. Také zde jsou interpretery.

doc – dokumentace (někdy `share/doc`).

games – hry a vzdělávací programy.

include – hlavičkové soubory pro jazyk C.

lib – knihovny, které nejsou nezbytně nutné pro jednouživatelský běh systému. Programy, které nejsou určeny ke spouštění uživatelem (případně v podadresářích). Read-only data různých aplikací, závislá na platformě (například moduly pro Perl a podobně).

local – adresář pro lokálně instalovaný software. Obsahuje podadresáře `bin`, `games`, `include`, `lib`, `sbin`, `share` a `src` s podobným významem, jako odpovídající položky pod `/usr`.

man – manuálové stránky (někdy `share/man`).

sbin – systémové programy, které nejsou nezbytně nutné pro běh systému (síťové služby, tiskový démon a podobně).

share – data nezávislá na architektuře (informace o časových zónách, `terminfo` a podobně).

spool – symbolický link na `/var/spool` z důvodů kompatibility.

src – zdrojové texty od systémových komponent.

tmp – symbolický link na `/var/tmp` z důvodů kompatibility.

Adresář `/var`

- Data, která se mohou měnit (tiskové fronty, mailboxy, lokálně generované fonty a podobně). Tento adresář není sdílitelný mezi počítači a může být na samostatném svazku.

adm – administrativní data. Často obsahuje systémové logy.

lock – aplikační zámky – například zámky na sériové linky.

log – systémové logy.

mail – poštovní schránky uživatelů (někdy ve `spool/mail`).

opt – modifikovatelná data pro balíky v `/opt`.

run – soubory, vztahující se k běžícím programům

spool – tiskové a poštovní fronty.

tmp – dočasné soubory.

Start systému – `init`

- **Program** – `/sbin/init`
- **Proces** – číslo 1.
- **System V** – řídící soubor `/etc/inittab`
- **BSD** – soubory `/etc/gettytab` a `/etc/rc`.

Jména `/sbin/init` a `/bin/sh` jsou jediná jména, zakompilovaná do kernelu. Zbytek systému má rozložení nezávislé na kernelu.

Úrovně běhu systému

- **Runlevels** – číslo od 0 do 6.
- Určuje, které subsystémy jsou aktivní.
- **Není v BSD**

- **0 – Halt** – zastavení systému.
- **1 – Single** – jednouživatelský běh systému.
- **2 – Multi** – víceuživatelský běh systému.
- **3 – Remote FS** – obvykle 2 + sdílení disků.
- **4 – Free.**
- **5 – Free** – RedHat zde má 3 + X-Window system.
- **6 – Reboot** – restart systému.

`init(8)`, `telinit(8)` . . . . . Změna stavu systému

```
# init [0123456aAbBcCsSqQ]
```

[**`0-6`**] Přechod na příslušnou úroveň chodu systému.
[**`a-c,A-C`**] Nastartování činností, které se však nikde neevidují.
[**`sS`**] Totéž co `init 1`, jen konzolou se stane současný terminál.
[**`qQ`**] Způsobí znovunačtení souboru `/etc/inittab`.

SysV Startovací skripty

- Startovací scripty v `/etc/init.d` – pro každý subsystém.
- Adresáře `/etc/rc[0-6].d`, linky `[SK][0-9][0-9]`*skript* (například `K56syslog` nebo `S60sshd`) do `../init.d`.
- Startovací skripty se spouštějí s parametrem `start` nebo `stop`.

RedHat-specifické vlastnosti:

- Adresář `/etc/sysconfig`.
- Parametry `restart` a `status` u startovacích skriptů.
- Program `chkconfig(8)`. Také na IRIXu.

Formát souboru `inittab`

- **Identifikace úlohy** – pozor, v některých systémech může být nejvýše dvouznaková.
- **Runlevel** – při které úrovni běhu systému se daná úloha má spouštět.
- **Způsob spouštění**
- **Příkaz**

Způsoby spouštění úloh

`once` – pouze jednou při přechodu na daný runlevel.
`respawn` – `init` úlohu restartuje po jejím skončení.
`sysinit` – pouze jednou při startu systému.
`wait` – pouze jednou při přechodu na daný runlevel. `init` čeká na dokončení úlohy.

Příklad souboru `inittab`

```
id:5:initdefault:
si::sysinit:/etc/rc.d/rc.sysinit
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
ud::once:/sbin/update
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
pf::powerfail:/sbin/shutdown -f -h +2
     "Power Failure; System Shutting Down"
pr:12345:powerokwait:/sbin/shutdown -c
     "Power Restored; Shutdown Cancelled"
1:12345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
4:2345:off:/sbin/mingetty tty4
x:5:respawn:/usr/bin/X11/xdm -nodaemon
```


Svazky

- **Soubor `/etc/fstab`** – seznam všech automaticky připojovaných svazků.
- **Soubor `/etc/mtab`** – seznam aktuálně připojených svazků. (Linux – také v `/proc/mounts`).
- **Program `mount(8)`** – připojení svazku.
- **Program `umount(8)`** – odpojení svazku.
- **Bind-mount** – připojení existujícího adresáře jako svazku (Linux). Též vícenásobné připojení téhož svazku.
- **Loop device** – Linux, Solaris. Vytvoření blokového zařízení ze souboru. Možnost připojení souboru jako svazku (např. ISO 9660 obraz CD).

Odkládací prostor

- **Disková oblast.** Někdy možnost swapovat do souboru.
- **Vzdálený odkládací prostor** – nevýhodné a těžké implementovat.
- **Seznam** – obvykle v `/etc/fstab`
- **Aktivace/deaktivace** – `swapon(8)`, `swapoff(8)`.
- **Další informace** – `swap(8)`, `/proc/swaps` v Linuxu.

Uživatelé a skupiny

- **UID/GID** – identifikace uživatele/skupiny z hlediska jádra systému.
- **Jméno uživatele** – používá se při přihlašování a u logování do souborů.

Soubor `/etc/passwd`

- **Základní databáze uživatelů**
- **Jméno uživatele** – klíč v `/etc/passwd`.
- **Heslo** – v zašifrované podobě.
- **UID** – číslo uživatele. Může být víc záznamů se stejným UID.
- **GID** – GID, které mají procesy po přihlášení. Obvykle 15-bitové.
- **GCOS** – komentář, obvykle celé jméno uživatele. Někdy několik záznamů oddělených čárkou, význam je systémově závislý (využívá jej např. `finger`). General Electric Comprehensive Operating System. Dennis Ritchie píše: *„Sometimes we sent printer output or batch jobs to the GCOS machine. The GCOS field in the password file was a place to stash the information for the* `$IDENTcard`*. Not elegant.“*
- **Domovský adresář** – pracovní adresář shellu po přihlášení.
- **Shell** – jméno programu, který se spustí po přihlášení.

Šifrování hesel

- **Standardně** – 25–krokový DES, 2 znaky sůl, zbytek heslo.
- **Knihovní funkce** – `crypt(3)`.
- **Jiné metody** – MD5, RC4, IDEA, DSS.

Ukládání hesel

- **Standardní UNIX** – hesla jsou vystavena útoku hrubou silou (John the Ripper) a slovníkovému útoku (`crack(8)`).
- **Shadow passwords** – hesla a další údaje jsou uloženy v souboru `/etc/shadow`. Omezení hesla na určitý čas, omezení frekvence změny hesla. Nutnost set-uid/gid u programů, pracujících s hesly.
- **BSD** – `/etc/master.passwd` – analogie shadow.
- **Trusted control base** – TCB – SCO, Digital UNIX – C2 bezpečnost. Nemožnost znovuvyužití hesla, volitelný zákaz změny hesla, volby vlastního hesla, atd. Nutnost set-uid/gid programů, pracujících s hesly.

Formát souboru `/etc/group`

- **Jméno skupiny** – identifikace pro logování do souboru a pro přepínání GID pomocí `newgrp(1)`.
- **Heslo skupiny** – obvykle nepoužito. Případné heslo může také být v `/etc/gshadow`. Použití hesla u systémů s GID-listem: Skupiny bez hesla jsou přidány do GID-listu hned po přihlášení, ostatní jen explicitně pomocí `newgrp(1)`.
- **Číslo skupiny** – 15-bitová identifikace pro systém.
- **Seznam uživatelů** – jména oddělená čárkami. Primární skupina je implicitně, uživatel zde nemusí být uveden.

Modifikace tabulky uživatelů

- **Speciální programy** – `vipw(8)` – je-li databáze uživatelů uložena i jinde (`shadow`, `master.passwd`).
- **Změna uživatelských informací** – `chfn(8)`.
- **Dávkové přidávání** – `useradd`, `groupadd`, `userdel`, `groupdel` – vytváří domovský adresář, alokuje volné UID, kopíruje soubory z `/etc/skel`. Problém: skupina pro každého uživatele?
- **`pwconv(8)`** – převod hesel do `shadow`.

Soubor `/etc/shells`

- Obsahuje jména souborů – na každém řádku jedno.
- Změna shellu pomocí `chsh(1)`.
- Některé služby jen pro uživatele s platným shellem.
- `/sbin/nologin` – shell pro pseudouživatele.

Name Service Switch

- **Alternativní zdroje dat** pro systémové tabulky (`passwd`, `group`, `hosts`, ...).
- **Implementace** – plug-iny do `libc`.
- **Konfigurace** – `/etc/nsswitch.conf`

```
passwd:         compat
group:          compat
hosts:          dns [!UNAVAIL=return] files
networks:       nis [NOTFOUND=return] files
```

- **Plug-iny** – Linux/glibc: `/lib/libnss_`⟨*služba*⟩`.so.X`.
- **Pozor na statické linkování** – Solaris nelze, Linux/glibc ano.

NSSwitch – konfigurace

- **Formát souboru** – databáze, mezera, popis služeb.
- **Podrobnější specifikace** – `[[!]`⟨*STATUS*⟩`=`⟨*AKCE*⟩ ...`]`
- **Možné akce**

`RETURN` – vrácení právě nalezené hodnoty nebo chyby.
`CONTINUE` – pokračovaní použitím další služby.

- **Možné návratové stavy**

`SUCCESS` – záznam nalezen, nedošlo k chybě. Implicitní akce je `RETURN`.
`NOTFOUND` – vyhledávání proběhlo bez chyby, ale záznam se nenašel. Implicitní akce je `CONTINUE`.
`UNAVAIL` – služba není trvale dostupná (např. nezkonfigurovaná). Implicitně `CONTINUE`.
`TRYAGAIN` – dočasná chyba (timeout, vyčerpání prostředků, atd.). Implicitně `CONTINUE`.

Pluggable Authentication Modules

- **PAM** – Sun Microsystems, nyní GPL nebo BSD. Hlavní vývoj nyní Red Hat.
- **Téměř všechny UNIXy** – distribuce Linuxu, Solaris, HP-UX. IRIX a některé BSD nikoliv. Různé stupně vývoje.
- **Modulární přístup k autentizaci** – čipové karty, hesla, biometriky, síťové databáze (Kerberos, LDAP, NIS), atd.
- **Architektura** – knihovna `libpam`, plug-iny v `/lib/security`, konfigurace v `/etc/pam.conf` a `/etc/pam.d/*`.

PAM – skupiny autentizace

`account` – jestli vůbec člověk má účet, nevyexpirované heslo, může k dané službě přistupovat?
`auth` – vlastní autentizace – ověření identity žadatele (heslo, jednorázové heslo, biometriky, čipové karty + PIN, atd.).
`password` – změny autentizačních mechanismů (změna hesla apod.).
`session` – akce před zpřístupněním služby a po ukončení (audit, připojení domovského adresáře, nastavení uživatelských limitů, atd).

PAM – formát konfigurace

```
auth     required   pam_securetty.so
auth     required   pam_env.so
auth     required   pam_nologin.so
auth     sufficient pam_unix.so nullok
auth     required   pam_deny.so
account  required   pam_unix.so
password required   pam_cracklib.so retry=3
password sufficient pam_unix.so nullok md5 shadow
password required   pam_deny.so
session  required   pam_limits.so
session  required   pam_unix.so
session  optional   pam_console.so
```

- **Řídící hodnoty:**

`required` – pokud selže, selže i celý autentizační proces.
`requisite` – totéž, ale skončí se hned.
`sufficient` – stačí k autentizaci bez ohledu na výsledek následujících modulů.
`optional` – spustí se, ale výsledek se použije pouze pokud jde o jediný modul daného typu.

Uživatelé a skupiny – programování

getpwnam(3), getpwuid(3) . . . . . Databáze uživatelů

```
#include <pwd.h>
#include <sys/types.h>
struct passwd *getpwnam(const char *name);
struct passwd *getpwuid(uid_t uid);

struct passwd{
  char *pw_name;
  char *pw_passwd;
  uid_t pw_uid;
  gid_t pw_gid;
  char *pw_gecos;
  char *pw_dir;
  char *pw_shell;
};
```

Vrací ukazatel na strukturu, popisující záznam daného uživatele. Pozor – funkce nejsou reentrantní, používá se vždy jedna a tatáž struktura.

Seznam všech uživatelů – `getpwent(3)`, `setpwent(3)`, `endpwent(3)`.

getgrnam(3), getgrgid(3) . . . . . Databáze skupin

```
#include <grp.h>
#include <sys/types.h>
struct group *getgrnam(const char *name);
struct group *getgrgid(gid_t gid);

struct group{
  char *gr_name;
  char *gr_passwd;
  gid_t gr_gid;
  char **gr_mem;
};
```

Vrací ukazatel na strukturu, popisující skupinu. Není reentrantní.

Seznam všech skupin – `getgrent(3)`, `setgrent(3)`, `endgrent(3)`.

Terminálové procesy

getty – inicializace linky, výpis zprávy, čekání na vstup.
login – načtení hesla, zápis do `wtmp` a `utmp`.
shell – uživatelský program.

Soubor `utmp`

- **`/var/run/utmp`**
- Seznam právě přihlášených uživatelů.
- **`#include <utmp.h>`**
- **`struct utmp`**, knihovní funkce pro procházení a modifikaci `utmp`.
- **Programy** – `who(1)`, `w(1)`.
- **Problém** – právo zápisu; set-uid/gid programy. Bezpečnostní riziko. Jiné řešení: knihovna a pomocný program – `utempter(8)`.

Soubor `wtmp`

- **`/var/log/wtmp`**
- Záznam o přihlášeních a odhlášeních uživatelů.
- Stejný formát jako u `utmp`, uživatel NULL značí odhlášení na daném terminálu.
- **Speciální záznamy** – start a ukončení systému, změna úrovně běhu systému. Změna systémového data.
- **Neexistující `wtmp`** – zákaz vedení záznamů. Při rotování `wtmp` nutno vždy vytvořit nový soubor.
- **Soubor `btmp`** – záznamy o chybných přihlášeních.
- **Programy** – `last(8)`, `lastb(8)`.

Správa zařízení v Linuxu

- **Sysfs** – od jádra 2.6.
- **Čistší implementace některých vlastností `/proc`**
- **Jednodušší implementace** – jeden soubor = jeden parametr.
- **Podadresáře pro** zařízení, sběrnice, třídy zařízení a další.

`/dev` a jádro 2.6

- **Klasický UNIX** – statický adresář `/dev`. Problémy: příliš mnoho nevyužitých souborů; pojmenování podle topologie (např. `/dev/dsk/c1t7d0s1`) nebo podle pořadí (`/dev/sdb3`).
- **DevFS** – Linux 2.4 (ale skoro nikdo nepoužívá), Solaris. Problémy: politika (pojmenování a přístupová práva) v kernel-space.
- **Linux 2.6** – `udev` – user-space devices. Jádro poskytuje informace, (přes sysfs), udev řeší politiku.
- **`/dev`** – obvykle jako `tmpfs`.
- **Možnosti pojmenování** – podle topologie, pořadí, sériového čísla, výrobce, driveru, atd.

◇ **Příklad:**

```
KERNEL="ttyUSB1", SYMLINK="pilot"
SYSFS{dev}="13:65", KERNEL="event*", \
    SYMLINK="input/keyboard_first"
BUS="usb", SYSFS{manufacturer}="Chicony", \
    SYSFS{idProduct}="0110", KERNEL="event*", \
    SYMLINK="input/keyboard_second"
```

- **Další informace:**
  `http://www.reactivated.net/writing_udev_rules.html`

Hotplug

- **Mechanismus jádra pro nově připojená zařízení.**
- **Spuštění externího programu** – `/sbin/hotplug`.
- **Reakce** – např. `udev` vytvoří soubor v `/dev` a podobně.
- **Coldplug** – spuštění hotplug událostí pro již existující zařízení v systému (například při bootu).

HAL

- **Hardware Abstraction Layer**
- **Jednotný přístup k zařízením různých typů** – atributy zařízení (např. typ: storage, camera).
- **Například pro storage zařízení** – `fstab-sync(8)`. Přidání zařízení do `fstab(5)`.
- **Sledování změn** – pokud zařízení neumí informovat o změně.
- **Výpis stávajícího stavu** – `lshal(8)`.

D-Bus

- **Desktop bus** – zasílání zpráv mezi různými subsystémy.
- **Obvykle dvě sběrnice** – pro systémové zprávy a pro každého uživatele s grafickým prostředím.
- **Možnost reakce na externí události** – např. okno s importem fotek z fotoaparátu. GUI reakce na události jádra.
- **Příklad** – `dbus_monitor(8)`.

Systémové procesy

Update

Stará se o zápis bufferů na disk. Volá každých 30 sekund službu jádra `sync(2)`. [Při vypínání počítače je dobré počkat 30 sekund.]

- **Nevýhoda** – špičková zátěž.
- **Lepší přístup** – postupná synchronizace.
- **Linux** – `kflushd`, `kupdate`.
- **Ostatní UNIXy** – podobné řešení.

Syslogd

- **Zpracovává hlášení o událostech**
- **Socket `/dev/log`** (`AF_UNIX`, `SOCK_DGRAM`).
- **Typ zprávy (facility):** `kern`, `user`, `mail`, `daemon`, `auth`, `syslog`, `lpr`, `news`, `uucp`, `cron`, `authpriv`, `local0` – `local7`.
- **Priorita zprávy (priority):** `emerg`, `alert`, `crit`, `err`, `warning`, `notice`, `info`, `debug`.

`openlog(3)` . . . . . Otevření systémového logu

```
#include <syslog.h>
void openlog(char *id, int option, int facility);
```

Řetězec `id` je připojen před každou zprávou. Parametr `option` je logický součet několika z následujících možností:

`LOG_CONS` – pokud se nepodaří odeslat zprávu, píše přímo na systémovou konzolu.
`LOG_NDELAY` – otevřít spojení ihned (jinak až při první zprávě).
`LOG_PERROR` – psát také na `stderr`.
`LOG_PID` – do zprávy zahrnout PID procesu.

`closelog(3)` . . . . . Uzavření logu

```
#include <syslog.h>
void closelog();
```

Ukončí zasílání zpráv (uvolní deskriptor).

`syslog(3)` . . . . . Zápis zprávy do logu

```
#include <syslog.h>
void syslog(int priority, char *fmt, ...);
```

Řetězec `fmt` má podobný význam jako v `printf(3)`, další argumenty jsou zpracovány v závislosti na něm.

`logger(1)` . . . . . Zápis do syslogu

```
$ logger [-is] [-p pri] [message ...]
$ logger -p lpr.notice -i Printer lp0 on fire!
```

- **Konfigurace** – `/etc/syslog.conf`.
- **Syntaxe:** *typ/priorita zprávy tabulátor soubor*.

Poslední položka se rozlišuje podle prvního znaku:

`/` – běžný soubor
`-` – totéž, nevolá se `fsync(2)`.
`|` – logování rourou do programu.
`*` – všem nalogovaným uživatelům.
`@` – logování po síti na jiný stroj.
ostatní – seznam uživatelů.

Příklad `/etc/syslog.conf`

```
kern.*                      /dev/console
*.info;mail.none;authpriv.none \
                            /var/log/messages
authpriv.*                  /var/log/secure
mail.*                      /var/log/maillog
*.emerg                     *
uucp,news.crit              /var/log/spooler
#*.debug                    -/var/log/debug
local0.info                 /var/log/ppp
local2.=info                |/usr/bin/log-parser
*.notice                    @loghost.domena.cz
```


Odstraňování chyb

- **Zjištění existence problému**
- **Určení místa, kde se problém vyskytuje**
- **Nalezení chyby**
- **Odstranění chyby** (je-li to možné :–).

Určení místa

- **Modularita UNIXu** – zjednodušuje určení místa výskytu problému.
- **Určení chybujícího programu** – například podle času přístupu ke spustitelnému souboru, k dynamickým knihovnám (nebo plug-inům), ke konfiguračním souborům.

Chybová hlášení

- **Zjistit, co chybové hlášení znamená**
- **Není-li to jasné** – přečíst dokumentaci, případně zdrojové texty programu.
- **Pokud není chybové hlášení** – zjistit, do kterého logovacího souboru se zapisuje.

- **Vyžádat si podrobnější informace** – zapnout podrobné výpisy v konfiguračním souboru nebo na příkazové řádce.
- **Zachytit podrobnější informace** – zapnout sledování zpráv priority `debug` v `syslog.conf`.

Sledování procesu

- **Služby jádra,** které proces postupně vykonává.
- **`strace(1)`** – Linux a další systémy.
- **`par(1)`** – IRIX.
- **`truss(1)`** – Solaris/SunOS.
- **Zjištění chyby služby jádra,** která vedla k ukončení procesu.

Dobrá rada na závěr

Po nějaké době řešení problému je dobré si znovu přečíst chybové hlášení a dokumentaci.

Cron

- **Vykonávání prací v zadaném čase.**
- **Úlohy běží pod UID/GID toho, kdo tuto úlohu požadoval.**
- **Standardní výstup a chybový výstup je zaslán uživateli e-mailem.**

`crontab(1)` . . . . . Pravidelně vykonávané úlohy

`-l` vypíše tabulku.
`-e` editace tabulky (`$EDITOR`, `$VISUAL`).
`-r` smaže tabulku.
`-u` ⟨*login*⟩ – nastavení pro jiného uživatele.
⟨*bez parametru*⟩ – bere tabulku ze std. vstupu.

Formát tabulky:

```
*/5 8-16 * * 1-5   /usr/bin/kdo-zrovna-pracuje
```

- **Položky** – minuta, hodina, den v měsíci, měsíc, den v týdnu, příkaz.

Logický součin podmínek. Pouze pokud den v měsíci a den v týdnu není `*`, bere se jejich logický součet.

Jednorázové úlohy

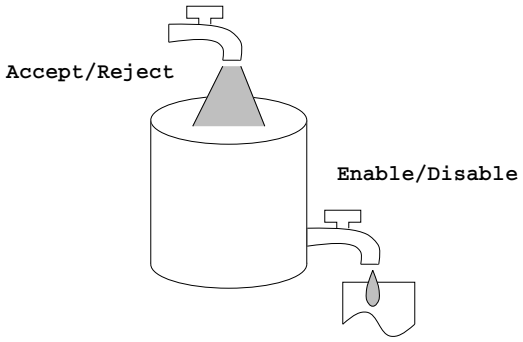
- **`at(1)`** – vykonání práce v zadaném čase. Pamatuje si proměnné prostředí.
- **`atq(1)`** – výpis fronty příkazů `at(1)`.
- **`atrm(1)`** – smazání úlohy z fronty.
- **`batch(1)`** – odložené vykonávání. Lépe použít samostatné dávkové systémy.
- **Spouštění** – většinou dělá `crond(8)`, někdy samostatný `atd(8)`.

Cron – řízení přístupu

`/etc/at.allow`
`/etc/at.deny`
`/etc/cron.allow`
`/etc/cron.deny`

Tiskárny

Přístup k tiskárně je řešen metodou *spoolingu*, tedy tiskových front. Požadavky na tisk se ukládají do fronty, odkud je obslužné programy vybírají a po případné úpravě posílají na fyzické zařízení.



BSD Tiskárny

- Jednodušší konfigurace.
- Soubor `/etc/printcap`.
- Podpora TCP/IP — číslo portu 515.
- Démon `lpd(8)`, `/etc/hosts.lpd`.
- Vstupní filtr, datový filtr, výstupní filtr.
- Příkaz `lpc(8)`.
- `lpr(1)`, `lpq(1)`, `lprm(1)`.

Konfigurace BSD tiskáren

```
dj|lp|HP DeskJet 540 - raw device:\
    :lp=/dev/lp1:\
    :sd=/var/spool/lpd/dj:\
    :sh:\
    :sf:\
    :mx#10000:\
    :lf=/var/log/lpd-errs:\
    :pl#60:
```



```
ps|HP DeskJet 540 - PostScript:\
    :lp=/dev/lp1:\
    :sd=/var/spool/lpd/ps:\
    :sh:\
    :sf:\
    :mx#10000:\
    :lf=/var/log/lpd-errs:\
    :if=/usr/local/sbin/psfilter:
netlj|HP LaserJet 4 over the Net:\
    :lp=/var/spool/lpd/netlj/.devnull:\
    :sd=/var/spool/lpd/netlj:\
    :rm=bigserver.foobar.com:\
    :rp=lj4:\
    :sh:\
    :sf:\
    :mx\#0:\
    :lf=/var/log/lpd-errs:
```

Příklad vstupního filtru

```
#!/bin/bash -
PRNAME="‘basename $0‘"
USER=nobody
MACHINE=‘uname -n‘
renice 9 $$ >/dev/null 2>&1
while [ "$1" != "" ] ; do
    case "$1" in
    -n)
        shift ; USER=$1
        ;;
    -h)
        shift ; MACHINE=$1
        ;;
    esac
    shift
done
```



```
gs -q -dSAFER -dNOPAUSE \
    -sDEVICE=cdj550 -sPAPERSIZE=a4 \
    -dBitsPerPixel=24 -dShingling=2 \
    -r300x300 -sOutputFile=- -
logger -p notice "lp-daemon: User $USER, \
    Machine $MACHINE, printer $PRNAME" >&2
exit 0
```

System V Tiskárny

- Složitější.
- Démon `lpsched(8)`.
- Programy `lpadmin(8)`, `lpshut(8)`,.
- `accept(8)`, `reject(8)`, `enable(8)`, `disable(8)`.
- `lpmove(8)`.
- `lp(1)`, `cancel(1)`, `lpstat(1)`.

Diskové kvóty (quotas)

- Sledování místa, obsazeného uživatelem.
- V rámci jednoho svazku.
- Kvóta na počet i-uzlů a bloků.
- Měkká/tvrdá kvóta, časový limit.
- Soubor `/quota`, `/quota.user`, `/quota.group` nebo přímo v metadatech FS.

`quota(1)` . . . . . Zjištění informací o kvótách

`$ quota [-v] [-u ⟨user⟩]`

Vypíše informace o překročených kvótách. S přepínačem `-v` vypisuje informace o všech kvótách. Volba `-u` – jen superuživatel.

`repquota(8)` . . . . . Informace o kvótách

Vypíše informace o kvótách všech uživatelů, označí překročené kvóty.

`quotactl(2)` . . . . . Nastavení kvót

Služba jádra, sloužící k práci s mechanismem kvót.

`quotaon(8)` . . . . . Zapnutí kvót

```
# quotaon [-a] [⟨filesystem⟩]
# quotaoff [-a] [⟨filesystem⟩]
```

Zapne/vypne kvóty pro daný systém souborů.

`quotacheck(8)` . . . . . Kontrola kvót

`# quotacheck [-a] [⟨filesystem⟩]`

Projde souborový systém a zkontroluje, jestli záznamy o kvótách souhlasí s aktuálním stavem.

`edquota(8)` . . . . . Editace kvót

`# edquota [-p ⟨user⟩] ⟨user⟩ ...`

Editace kvót pro jednotlivé souborové systémy. S přepínačem `-p` provádí dávkovou editaci podle vzoru. Proměnná `EDITOR`.

Sítě TCP/IP

- **internet** – několik sítí propojených dohromady.
- **Internet** – celosvětová síť s protokolem TCP/IP.
- **Transmission Control Protocol/Internet Protocol**
- **RFC** – Request For Comments – dokumenty, popisující jednotlivé protokoly. `ftp.fi.muni.cz:/pub/rfc`. Některé z nich jsou standardy
- **IETF** – Internet Engineering Task Force. Internet drafts – navrhy protokolu. `ftp.fi.muni.cz:/pub/internet-drafts`. Omezená časová platnost.

Historie Internetu

- **1969** ARPANET – první síť s přepojováním packetů – 4 uzly.
- **1977** Začíná vývoj protokolů TCP/IP (Stanford, BBN, University College London).
- **1980** Provoz TCP/IP v ARPANETu. UCB implementuje TCP/IP pro BSD.
- **1983** TCP/IP se stává standardem v ARPANETu. Sun Microsystems – TCP/IP v komerční sféře.
- **1985** NSFnet – jádro současného Internetu.

Charakteristiky TCP/IP

- Internet – síť s *přepojováním packetů* (opozitum k *přepojování okruhů*; kombinace obou je ATM).
- Směrovač (router) – počítač, zajišťující propojení více sítí.
- Stejné protokoly pro všechny sítě a druhy sítí.
- Směrování na základě cílové sítě, nikoli cílové adresy.

Internet Protocol verze 4

- **Datagram** (packet) – balík dat omezené velikosti.
- **Směrování** – každý packet je směrován nezávisle na ostatních.
- **Nespolehlivost** – datagram může dojít vícekrát nebo se ztratit. datagramy mohou změnit pořadí.
- **Fragmentace packetů** – při průchodu do rozhraní s menší maximální velikostí datagramu (MTU).

Vrstvy internetových protokolů

Nižší vrstvy přibližně podle OSI:

- **Fyzická** – například UTP kabel, optické vlákno, metalický okruh.
- **Linková** – síťové rozhraní (Ethernet, ATM LANE, PPP, HDLC, FDDI).
- **Síťová** – IP (datagramy), ARP (získání HW adresy), RARP (získání vlastní IP adresy), IPv6.
- **Transportní** – ICMP (řídící zprávy), IGMP (skupinová adresace), TCP (proudy dat), UDP (datagramy), ICMPv6.
- **Aplikační** – protokoly jako SMTP, FTP, telnet, NTP, SNMP, ...

Adresování v IPv4

- **Internet Protocol** – verze 4, RFC 791.
- **IP adresa** – 4 bajty. Zapisuje se jako 4 dekadická čísla, oddělená tečkami.

Třídní adresace

- **Adresa typu A** – `0.x.x.x`–`127.x.x.x`.
- **Adresa typu B** – `128.0.x.x`–`191.255.x.x`.
- **Adresa typu C** – `192.0.0.x`–`223.255.255.x`.
- **Adresa typu D** – `224.0.0.0`–`239.255.255.255` (`mcast.net`).
- **Adresa typu E** – `240.0.0.0`–`255.255.255.255` – rezerva.

Beztřídní adresace

- **Nyní** – nedostatek adres, vyčerpání adresního prostoru.
- **CIDR** Prefix adresy (adresa, maska). RFC 1518.

Adresy uvnitř sítě

- **Interface** – síťové rozhraní – má svoji IP adresu (`147.251.48.18`).
- **Netmask** – maska sítě – určuje, které adresy jsou na téže síti (`255.255.255.0`).
- **Network address** – adresa sítě (*interface* `and` *netmask* – `147.251.48.0`).
- **Broadcast address** – adresa pro všesměrové vysílání uvnitř sítě (*interface* `or not` *netmask*).
- **My address** – doplní se do adresy odesílatele, nezná-li odesílatel svoji adresu (`0.0.0.0`).
- **Limited broadcast** – všesměrové vysílání pro tuto síť – neprochází přes routery (`255.255.255.255`).
- **Loopback address** – pro adresování sebe sama (`127.0.0.1`)
- **Privátní sítě** – `10.0.0.0/8`, `172.16.0.0/12`, `192.168.0.0/16` (RFC 1918). Nesmí se objevit v Internetu.

◇ **Úkol:** Jaká je numericky nejvyšší adresa v bloku `172.16.0.0/12`?

Formát IP packetu

- **Verze protokolu** – 4 bity.
- **Header length** – 4 bity, značí počet 32-bitových jednotek.
- **Type of service** – 3 bity priorita, 1 bit latence, 1 bit propustnost, 1 bit bezeztrátovost. Zarovnáno na 8 bitů.
- **Total length** – 16 bitů.
- **Identification** – 16 bitů – k fragmentaci.
- **Flags** – 3 bity – Don't fragment, More fragments, 1 bit rezervovaný.
- **Fragment offset** – 13 bitů, značí násobek osmi bajtů.
- **Time to live** – TTL, 8 bitů. Životnost datagramu v sekundách. Každý router musí TTL snížit aspoň o jedničku.
- **Protocol** – 8 bitů. Označení vyšší vrstvy, které datagram patří.
- **Header checksum** – 16 bitů.
- **Source IP address** – 32 bitů.
- **Destination IP address** – 32 bitů.
- **Options** – délka je násobek 32 bitů.
- **Data** – délka je násobek 32 bitů, max. 65536 bajtů.

Fragmentace packetů

- Prochází-li datagram do sítě s menší MTU.
- Minimální MTU je 576 bajtů.
- Všechny fragmenty mají stejné *identification*.
- Fragmenty mají *more fragments* flag nastaven na 1.
- Poslední fragment a nefragmentované packety mají tento flag nastavený na 0.
- Fragmenty (kromě prvního) mají nenulový *fragment offset*.
- **Znovusestavení datagramu** – smí provádět pouze cílový počítač. Někdy je též prováděno hraničním routerem nebo firewallem privátní sítě.
- **Path MTU discovery** – zjištění MTU cesty – posílá se nefragmentovatelný packet, sledují se odpovědi.

Volitelné položky v IP datagramu

- **Record route** – trasování cesty datagramu.
- **Loose source route** – minimální cesta datagramu.
- **Strict source route** – úplná cesta datagramu.
- **Timestamp** – trasování s časovým razítkem.
- **Security** – stupeň utajení datagramu.

Internet Protocol version 6

- **Větší adresní prostor.**
- **Mobilita** – práce ve více sítích, přechod mezi sítěmi za běhu aplikací, domovský agent.
- **Zabezpečení** – šifrované a podepisované packety – protokol IPSEC.
- **Autokonfigurace** – zjištění informací o síti přímo ze sítě.
- **Více adres** – každé rozhraní může mít/má více adres.
- **Způsoby přenosu** – unicast, multicast, anycast.
- **Více IPv6 adres** na jedno rozhraní.

Adresace v IPv6

- **128-bitová adresa**
- **Obvyklý zápis** – čtveřice šestnástkových čísel.
- **Příklad:** `3ffe:ffff:0000:f101:0210:a4ff:fee3:9562`
- **Vypuštění úvodních nul** – `3ffe:ffff:0:f101:210:a4ff:fee3:9562`
- **Vypuštění sekvence `0000`** – `3ffe:ffff::f101:210:a4ff:fee3:9562`.
- **CIDR** – každá adresa má k sobě masku sítě, specifikuje se lomítkem a počtem bitů –`3ffe:ffff::12/64`.

Formát IPv6 packetů

- **Hlavička pevné délky,** řetězení hlaviček.
- **Verze protokolu** – 4 bity, hodnota vždy 6.
- **Priorita** – 8 bitů, třída provozu.
- **Identifikace toku** – 20 bitů.
- **Délka packetu** – 16 bitů.
- **Next header** – 8 bitů (identifikace další hlavičky, např. vyšší vrstvy).
- **Hop limit** – 8 bitů (ekvivalent TTL u IPv4).
- **Zdrojová adresa** – 128 bitů.
- **Cílová adresa** – 128 bitů.

Přechodové mechanismy

- **Dual stack** – podpora IPv4 a IPv6 v jednom počítači.
- **Tunelování** – zapouzdření IPv6 packetů do IPv4 (protokol SIT, 41).
- **Autotunelování** – (6to4) – adresy `2002:xxxx:yyyy:/48`, vytvořené z IPv4 adresy. Schování bloku `/48` IPv6 adres za jednu IPv4 adresu. Komunikace do nativního IPv6 internetu přes 6to4 relay 192.88.99.1 (2002:c058:6301::).

Speciální IPv6 adresy

- **Loopback** – `::1` (ekvivalent `127.0.0.1` v IPv4).
- **Nespecifikovaná adresa** – `::` (ekvivalent `0.0.0.0` v IPv4).
- **Lokální adresa linky** – `fe80::/10`
- **Site-local address** – `fec0::/10` – ekvivalent privátních adres dle RFC1918, nyní zastaralé (RFC 3879).
- **Adresy pro 6bone** (testovací síť pro) IPv6 – prefix `3ffe::/16` (zastaralé
- **Adresy pro příklady** – `3ffe:ffff::/32`.
- **IPv4 kompatibilní adresy** – `::/96`.
- **IPv4 mapované adresy** – `::ffff:0:0/96`.

◇ **Úkol:** Jaká je numericky nejvyšší link-local adresa?

EUI-64 formát adresy

- **EUI-64 formát adresy** – lokální část se odvozuje z fyzické adresy.
- **EUI-64 pro ethernet** – MAC adresa, uprostřed vloženo `fffe`, 7. nejvyšší bit nastaven na 1 pro unicast.
- **Příklad** – MAC adresa `00:D0:B7:6B:4A:B2`, prefix sítě `fe80::/10`, pak IPv6 addresa je `fe80::2d0:b7ff:fe6b:4ab2`.
- **Usnadňuje autokonfiguraci** – směrovač vysílá *router advertisement*, kde je uveden `/64` prefix lokální sítě. Viz též `radvd(8)`.

◇ **Úkol:** Jaká by byla EUI-64 adresa počítače `aisa.fi.muni.cz`, je-li adresní prefix `2001:718:801:230::/64`?

Specifické vlastnosti IPv6

- **Fragmentace packetů** – není. Vysílající musí dělat *path MTU discovery*. Fragmentace popsána v samostatná *next header*.
- **Spolupráce s linkovou vrstvou** – NDP (neighbour discovery protocol). Náhrada ARP. Zjišťování adresního prefixu sítě, směrovacích informací, atd.

Odkazy

- **Linux and IPv6 howto** – *http://www.bieringer.de/linux/IPv6/ /IPv6-HOWTO/IPv6-HOWTO.html*
- **USAGI Project** – *http://www.linux-ipv6.org* – UniverSAl playGround for Ipv6.
- **KAME Project** – *http://www.kame.net/* – implementace IPv6 pro BSD systémy.

IP nad Ethernetem

Formát rámce Ethernet v2

- **Cílová adresa** – 6 bajtů.
- **Zdrojová adresa** – 6 bajtů.
- **Typ** (protokol vyšší vrstvy) – 2 bajty.
- **Data** – 46–1500 bajtů.
- **CRC** – 4 bajty.

Způsoby propojení sítí

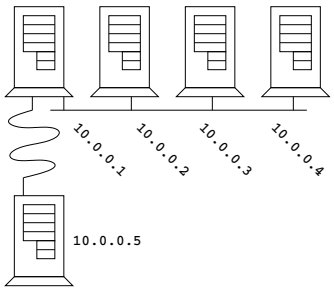
- **Repeater** – v podstatě propojení na fyzické úrovni.
- **Bridge/Switch** – propojení na linkové úrovni.
- **Router** – propojení na síťové úrovni. Pro každou třídu protokolů může být zvláštní router. Nezávislost na ethernetu.

ARP/RARP

- **Převod HW adres na IP adresy a naopak**
- **HW type** – 2 bajty (1 = Ethernet).
- **Protocol type** – 2 bajty (0x0800 = IP).
- **HW length** – 1 bajt (Ethernet = 6).
- **Protocol length** – 1 bajt (IP = 4).
- **Operation** – 2 bajty (1 = ARP request, 2 = ARP response, 3 = RARP request, 4 = RARP response).
- **Sender layer 2 addr**
- **Sender layer 3 addr**
- **Target layer 2 addr**
- **Target layer 3 addr**

- **ARP cache** – počítače si pamatují ARP informace po určitou dobu.
- **Proxy ARP** – namapování více IP adres na jednu HW adresu za účelem jednoduchého směrování (viz PPP).
- **Pozor!** starší systémy neumožňovaly mít více IP adres přiřazených jedné HW adrese.

Proxy ARP



ICMP

- **Internet Control Message Protocol.**
- Chybové a řídící zprávy IP protokolu.
- ICMP zpráva – uvnitř IP datagramu.
- Podává se původnímu odesílateli příslušného datagramu.

Formát ICMP zpráv

- **Type** – 8 bitů – typ ICMP zprávy.
- **Code** – 8 bitů – přídavný kód.
- **Checksum** – 16 bitů – kontrolní součet ICMP zprávy.

ICMP Echo Request/Echo

- Pro testování dostupnosti počítače.
- Každý počítač v síti je povinen na ICMP echo request (*type* = 8) odpovědět ICMP echo (*type* = 0) se stejnou datovou částí.
- Další položky: 16 bitů identifikace, 16 bitů číslo sekvence.

Neodsažitelnost adresáta

- **ICMP destination unreachable** – type 3.
- **Code** obsahuje bližší informace.
- Datagram dále obsahuje hlavičku a prvních 64 bitů IP datagramu.

Typy zpráv v závislosti na *code*:

- **Network unreachable**
- **Host unreachable**
- **Protocol unreachable**
- **Port unreachable**
- **Fragmentation needed**
- **Source route failed**
- **Destination network unknown**
- **Destination host unknown**

Zahlcení routeru

- **ICMP source quench**
- Type = 4, code = 0, dále hlavička a 64 bitů ze začátku IP datagramu.
- Vysílající strana musí reagovat snížením toku dat.

Přesměrování datagramů

- **ICMP redirect** – type = 5.
- Žádost o přesměrování dalších packetů na jiný router.
- Datagram obsahuje adresu nového routeru, IP hlavičku a prvních 64 bitů dat IP datagramu.
- **Code** určuje typ přesměrování: Přesměrování všech datagramů pro síť nebo pro jeden počítač, případně přesměrování služby pro síť nebo pro jeden počítač.

◇ **Úkol:** Jak pozná router, že má poslat ICMP redirect?

Překročení času

- **ICMP time limit exceeded** – type = 11.
- Životnost datagramu (TTL) vypršela, nebo byl překročen čas na sestavení datagramu z fragmentů.
- Datagram obsahuje IP hlavičku a prvních 64 bitů dat IP datagramu.

Ostatní problémy

- **ICMP parameter problem** – type = 12.
- **Pointer** – 8 bitů, zarovnáno na 32 bitů.
- Datagram dále obsahuje IP hlavičku a prvních 64 bitů dat IP datagramu.
- Je-li *code* = 1, obsahuje *pointer* index na místo datagramu, které způsobilo problém. Je-li *code* = 0, nemá položka *pointer* význam.

Další ICMP zprávy

- **Timestamp request** (type = 13).
- **Timestamp reply** (type = 14).
- **Information request** (type = 15) – žádost o adresu sítě.
- **Information reply** (type = 16).
- **Address mask request** (type = 17).
- **Address mask reply** (type = 18).

ICMPv6

- **Analogie ICMP pro IPv6**
- **Protokol číslo 58**

Formát ICMPv6 packetů

- **Type** – 8 bitů, nejvyšší bit odlišuje chybové a informativní zprávy.
- **Code** – 8 bitů, bližší určení.
- **Checksum** – 16 bitů.

Chybové zprávy

- **1** Destination unreachable.
- **2** Packet too big.
- **3** Time exceeded.
- **4** Parameter problem.

Informativní zprávy – I.

- **128** Echo request.
- **129** Echo reply.
- **130** Group Membership Query.
- **131** Group Membership Report.
- **132** Group Membership Reduction.
- **133** Router Solicitation.
- **134** Router Advertisement.
- **135** Neighbor Solicitation.
- **136** Neighbor Advertisement.
- **137** Redirect.
- **138** Router Renumbering.
- **139** ICMP Node Information Query.
- **140** ICMP Node Information Response.
- **141** Inverse Neighbor Discovery Solicitation Message.
- **142** Inverse Neighbor Discovery Advertisement Message.

Informativní zprávy – II.

- **143** MLDv2 Multicast Listener Report.
- **144** Home Agent Address Discovery Request Message.
- **145** Home Agent Address Discovery Reply Message
- **146** Mobile Prefix Solicitation.
- **147** Mobile Prefix Advertisement
- **148** Certification Path Solicitation.
- **149** Certification Path Advertisement.
- **150** Experimental mobility protocols.
- **151** Multicast Router Advertisement.
- **152** Multicast Router Solicitation.
- **153** Multicast Router Termination.

UDP

- **User Datagram Protokol**
- **Nespojovaná transportní služba**
- **Porty** – rozlišení mezi více adresáty (a zdroji) v rámci jednoho počítače. Port je 16-bitové celé číslo.
- **Well-known ports** – porty, na kterých lze očekávat obecně známé služby.

Formát UDP rámce

- **Source port** – 16 bitů.
- **Destination port** – 16 bitů.
- **Length** – 16 bitů.
- **Checksum** – 16 bitů. Nepovinný. Součet UDP hlavičky a IP pseudohlavičky.

IP pseudohlavička:

- Zdrojová IP adresa, cílová IP adresa.
- Nula – 8 bitů.
- Protokol – 8 bitů.
- Délka packetu – 16 bitů.

TCP

- **Transmission Control Protocol**
- **Spojovaná služba**
- **Spolehlivá služba**
- **Duplexní proud dat**
- **Buffering**
- **Porty** – podobně jako v UDP.

Protokol TCP zaručuje následující:

- Správné pořadí datagramů.
- Duplicitní datagramy jsou vyřazeny.
- Potvrzování přenosu dat.
- Opakování přenosu, nedojde-li potvrzení.

TCP spojení

- **Zdrojová IP adresa**
- **Zdrojový TCP port**
- **Cílová IP adresa**
- **Cílový TCP port**

Vytvoření spojení

- **Klient-server.**
- **Server** – *poslouchá* (listen) – čeká na spojení na určitém portu a je ochoten *přijmout* (accept) spojení.
- **Klient** – *spojuje* se (connect) na určitou službu (port) cílového stroje. Spojení může navazovat i z konkrétního zdojového portu.
- **Navázání spojení** – *three-way handshake*:

Klient		Server
SYN (seq $= x$)	$\longrightarrow$	
	$\longleftarrow$	SYN-ACK
ACK (ack $= y + 1$)	$\longrightarrow$	(seq $= y$, ack $= x + 1$)

- **Číslo sekvence** – každý packet obsahuje 32-bitové číslo, udávající pořadí dat v něm v rámci TCP spojení. Při otevírání spojení si oba konce stanoví počáteční číslo sekvence. *Bezpečnost!*

- **Klouzající okno** – vysílající strana je oprávněna vysílat další packet(y) bez nutnosti potvrzení předchozího packetu. Velikost okna je dohodnuta při otevírání spojení.
- **Potvrzování** – ACK packet, piggybacking při duplexním spojení.
- **Změna velikosti okna** – každý ACK packet obsahuje počet slov, který je druhá strana ochotna přijmout.
- **Out-of-band data** – urgentní data, zasílaná mimo pořadí.

Formát TCP rámce

- **Source port** – 16 bitů.
- **Destination port** – 16 bitů.
- **Sequence number** – 32 bitů.
- **Acknowledgement number** – 32 bitů.
- **Header length** – 4 bity – počet 32-bitových slov, velikost hlavičky včetně volitelných položek.
- **Window** – 16 bitů. Počet slov, které je odesílatel schopen přijmout.

Formát TCP rámce II.

- **Checksum** – 16 bitů – kontrolní součet včetně pseudozáhlaví (viz UDP).
- **Urgent pointer** – poslední bajt urgentních dat.
- **URG** – 1 bit. Doručit tento segment co nejrychleji. Položka *Urgent pointer* je platná.
- **SYN** – 1 bit. Synchronizace sekvenčních čísel; žádost o zřízení spojení.
- **ACK** – 1 bit. Položka *Acknowledgement number* je platná.
- **RST** – 1 bit. Požadavek na reset spojení („Connection reset by peer“) – posílá se jako odpověď na packet bez SYN flagu, který nepřísluší žádnému existujícímu spojení.
- **PSH** – 1 bit (push). Požadavek na rychlé doručení tohoto segmentu vyšší vrstvě sítě.
- **FIN** – 1 bit. Ukončení spojení – odesílatel vyslal všechna data.
- **Options** – zarovnáno na 32 bitů – volitelné položky (např. maximální velikost segmentu, který je odesílatel schopen přijmout, atd.).

Programování síťových aplikací

- **Několik druhů API**
- **BSD Sockets** – de facto standard, nejpoužívanější.
- **Streams** – pochází z UNIXu System V (např. Lachman TCP/IP – SCO UNIX).

BSD Sockets API

- **API pro meziprocesovou komunikaci** – někdy nazývané BSD IPC (oproti SysV IPC = semafory, fronty zpráv a sdílená paměť).
- **Nezávislé na síťovém protokolu** – je možné provozovat nad různými *rodinami protokolů* (`PF_UNIX`, `PF_INET`, `PF_AX25`, `PF_IPX`, `PF_APPLETALK`, `PF_BRIDGE`, `PF_NETROM`, `PF_AAL5`, `PF_X25`, `PF_INET6` – citováno z Linuxu).
- **Socket** – schránka – koncový komunikační uzel. Jeden konec síťového spojení. Deskriptor.
- **Rozšíření abstrakce souboru** – sémantika podobná běžným souborům; použití `read(2)`, `write(2)`) nebo speciální služby pro sockety (`sendmsg(2)`, `recvmsg(2)`, atd.).

Síťový formát dat

- **Komunikace mezi různými stroji** – nutnost stanovit pořadí bajtů v 16-bitovém a 32-bitovém čísle.
- **Síťový formát dat** – big endian.
- **Nativní formát dat** – little-endian u ia32, ia64, x64, AXP, ARM; big-endian u SPARC, HP-PA a dalších. Volitelné u MIPS, PPC, SPARCv9.

Práce se síťovým formátem dat

```
#include <netinet/in.h>
unsigned long htonl(unsigned long hostl);
unsigned long ntohl(unsigned long netl);
unsigned short htons(unsigned short hosts);
unsigned short ntohs(unsigned short nets);
```


`socket(2)` . . . . . Vytvoření socketu

```
#include <sys/types.h>
#include <sys/socket.h>
int socket(int domain, int type, int proto);
```

Vytvoří socket a vrátí jeho deskriptor.

`domain` – rodina adres – způsob komunikace přes socket. Odpovídá rodině protokolů. `AF_UNIX`, `AF_INET`, atd.

`type` – sémantika komunikace. Viz dále.

`proto` – protokol. Obvykle existuje pro každou kombinaci (`domain`, `type`) nejvýše jeden. Pak zde může být 0. Viz `/etc/protocols`.

Typy socketů

Položka `type` určuje chování socketu a jeho schopnosti.

`SOCK_STREAM` – plně duplexní spolehlivý uspořádaný proud dat, případně podporuje i posílání dat mimo pořadí (out-of-band data).

`SOCK_DGRAM` – datagramová služba.

`SOCK_RAW` – přímý přístup na síťové zařízení. Povoleno jen superuživateli.

`SOCK_SEQPACKET` – uspořádané spolehlivé duplexní spojení pro přenost packetů do jisté maximální délky. Může být požadováno načtení packetu jedním `read(2)` nebo podobnou službou.

`socketpair(2)` . . . . . Dvojice socketů

```
#include <sys/types.h>
#include <sys/socket.h>
int socketpair(int domain, int type, int proto,
    int sd[2]);
```

Vrátí nepojmenovanou dvojici vzájemně spojených socketů. Tyto sockety (`sd[0]` a `sd[1]`) nelze rozlišit.

`getprotoent(3)` . . . . . Získání čísla protokolu

```
#include <netdb.h>
struct protoent *getprotoent();
struct protoent *getprotobyname(char *name);
struct protoent *getprotobynumber(int proto);
void setprotoent(int stayopen);
void endprotoent();

struct protoent {
    char *p_name;
    char **p_aliases;
    int p_proto;
}
```

Funkce slouží ke čtení tabulky protokolů v souboru `/etc/protocols`. Parametr `stayopen` říká, má-li být soubor `protocols` otevřen i během volání `getprotobyname(2)` a `getprotobynumber(2)`.

Tyto funkce jsou nereentrantní – nelze je použít v multithreadovém prostředí ani uvnitř ovladače signálu.

Příklad souboru `/etc/protocols`

```
ip   0   IP   # internet protocol
icmp 1   ICMP # internet control message p.
igmp 2   IGMP # internet group multicast p.
ggp  3   GGP  # gateway-gateway p.
tcp  6   TCP  # transmission control p.
pup  12  PUP  # PARC universal packet p.
udp  17  UDP  # user datagram p.
raw  255 RAW  # RAW IP interface
```

◇ **Úkol:** Napište programy `getprotobyname` a `getprotobynumber`, které na standardní vstup vypíší číslo protokolu na základě jména a naopak.

`bind(2)` . . . . . Pojmenování socketu

```
#include <sys/types.h>
#include <sys/socket.h>
int bind(int fd, struct sockaddr *addr,
    int addrlen);
```

Přiřadí existujícímu socketu adresu v příslušné rodině adres. Doména `AF_UNIX` – adresou je cesta k souboru (`/dev/log`), v `AF_INET` je adresou dvojice (číslo portu, IP adresa).

Adresace socketů

Obecná adresa:

```
struct sockaddr {
    u_short sa_family;
    char sa_data[14];
};
```

Adresa v doméně `AF_UNIX`:

```
struct sockaddr_un {
    u_short sun_family;
    char sun_path[UNIX_PATH_MAX];
};
```

Adresa v doméně `AF_INET`:

```
struct sockaddr_in {
    u_short sin_family;
    u_short sin_port;
    u_long sin_addr;
    char sin_zero[8];
};
```


Pojmenování portů

- **Číslo portu** – 16 bitů.
- **Privilegované porty** – 0–1023.
- **Známé služby** – well known services.

`getservbyname(3)` . . . . . Získání čísla služby

```
#include <netdb.h>
struct servent *getservbyname(char *name,
    char *proto);
struct servent *getservbyport(int port,
    char *proto);
void setservent(int stayopen);
struct servent *getservent();
void endservent();
```

Tyto rutiny slouží ke čtení tabulky `/etc/services`. Jsou analogické funkcím pro čtení souboru `/etc/protocols`.

Struktura `servent` vypadá následovně:

```
struct servent {
    char *s_name;
    char **s_aliases;
    int s_port;
    char *s_proto;
};
```

Příklad souboru `/etc/services`

```
discard    9/tcp  sink null
discard    9/udp  sink null
chargen   19/tcp  ttytst source
chargen   19/udp  ttytst source
ftp-data  20/tcp
ftp       21/tcp
fsp       21/udp  fspd
telnet    23/tcp
smtp      25/tcp  mail
time      37/tcp  timserver
```


Jména a adresy

- **Převod IP adres na jména a naopak** – `/etc/hosts`, NIS/YP, DNS.
- Jednomu jménu může být přiřazeno více adres.
- Jeden počítač (rozhraní, adresa) může mít více jmen.
- **Resolver** – mechanismus převodu jmen na IP adresy a naopak.

`gethostbyname(3)` . . . . . Převod jména na IP adresu

```
#include <netdb.h>
extern int h_errno;
struct hostent *gethostbyname(char *name);
struct hostent *gethostbyaddr(char *addr, int len,
    int type);
void sethostent(int stayopen);
struct hostent *gethostent();
void endhostent();
void herror(char *s);
```

Převod jména na adresu a nazpět. Příslušné funkce jsou nereentrantní. Resolver lze volat i jinak – funkce `res_query(3)` a další.


```
struct hostent {
    char    *h_name;
    char    **h_aliases;
    int     h_addrtype;
    int     h_length;
    char    **h_addr_list;
};
```

Příklad souboru `/etc/hosts`

```
127.0.0.1      localhost localhost.localdomain
147.251.50.60  pyrrha pyrrha.fi.muni.cz
```

◇ **Úkol:** Napište program, který pomocí `gethostbyname(3)` vypíše IP adresy a všechny aliasy k zadanému jménu.

getsockname(2) . . . . . Zjištění jména socketu

```
#include <sys/types.h>
#include <sys/socket.h>
int getsockname(int s, struct sockaddr *name,
    int *namelen);
```

Vrátí jméno – v AF_INET dvojice (IP adresa, port) – zadaného socketu.

◇ **Úkol:** Napište program, který zjistí, je-li na jeho standardním vstupu socket. Pokud ano, vypíše jeho adresu. Napište program, který vytvoří pojmenovaný socket (AF_UNIX) a výše uvedenému programu jej předá jako standardní vstup.

listen(2) . . . . . Čekání na spojení

```
#include <sys/socket.h>
int listen(int sock, int backlog);
```

Pojmenovaný socket může čekat na příchozí spojení pomocí listen(2). Parametr backlog definuje maximální počet příchozích spojení, které jsou ve frontě. Další jsou pak odmítnuty s chybou ECONNREFUSED. Pouze pro sockety typu SOCK_SEQPACKET a SOCK_STREAM.

accept(2) . . . . . Přijetí spojení na socketu

```
#include <sys/types.h>
#include <sys/socket.h>
int accept(int sock, struct sockaddr *addr,
    int *addrlen);
```

Přijme spojení, čekající ve frontě. Vrátí nový deskriptor, odpovídající spojení. Pouze pro SOCK_STREAM a SOCK_SEQPACKET. Chceme-li neblokující accept(2), je možné použít na poslouchací socket select(2) pro čtení. Parametr addr po návratu obsahuje adresu druhého konce socketu.

◇ **Úkol:** Napište program netread, který dostane jako parametr jméno nebo číslo protokolu a jméno nebo číslo portu, otevře příslušný port a přijme na něm spojení. Na standardní výstup vypíše adresu a port, ze které obdržel spojení, dále vše co přečte ze socketu a pak skončí. Vyzkoušejte funkčnost příkazem telnet ⟨*stroj*⟩ ⟨*port*⟩.

connect(2) . . . . . Navázání spojení

```
#include <sys/types.h>
#include <sys/socket.h>
int connect(int sockfd, struct sockaddr
    *server, int addrlen);
```

Pro SOCK_DGRAM určuje, ze které adresy je ochoten socket přijímat data a na kterou adresu posílá data. Žádné další akce nejsou v nižších vrstvách protokolu provedeny.

Pro SOCK_STREAM a SOCK_SEQPACKET se systém pokusí spojit na vzdálený socket, určený pomocí parametru server.

◇ **Úkol:** Napište program netwrite, který si otevře socket a spojí se protokolem TCP na danou adresu a port. Po ustavení spojení zapíše vše co přečte ze standardního vstupu do socketu a pak uzavře spojení.

getpeername(2) . . . . . Adresa druhého konce socketu

```
#include <sys/socket.h>
int getpeername(int s, struct sockaddr *name,
   socklen_t *namelen);
```

Vrátí adresu socketu, se kterým je socket s spojený. Není-li socket spojený, vrátí –1 a errno = ENOTCONN.

Práce s IP adresami

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int inet_pton(int af, const char *cp,
    void *addrp);
const char *inet_ntop(int af, const void *src,
    char *dst, socklen_t length);
```

Převod textového (číselného) zápisu adresy na struct in_addr nebo struct in6_addr a naopak.

Starší funkce jen pro IPv4: inet_ntoa(3) a inet_aton(3).

Nezávislost na síťovém protokolu

getaddrinfo(3) . . . . . Překad síťových adres a služeb

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>

int getaddrinfo(const char *node,
     const char *service,
     const struct addrinfo *hints,
     struct addrinfo **result);
void freeaddrinfo(struct addrinfo *result);
const char *gai_strerror(int retval);
```

Funkce nahrazuje getservbyname(3) nebo getservbyport(3), a dále gethostbyname(3) nebo gethostbyaddr(3) nebo getipnodebyname(3) nebo getipnodebyaddr(3). Je reentrantní a vrací struktury přímo použitelné v connect(2) nebo bind(2).

Vrací dynamicky alokovanou strukturu (zřetězený seznam), kterou lze uvolnit z paměti pomocí freeaddrinfo(3).

Dojde-li k chybě (návratová hodnota není nula), lze získat popis chyby pomocí gai_strerror(3).


```
struct addrinfo {
     int     ai_flags;
     int     ai_family;
     int     ai_socktype;
     int     ai_protocol;
     size_t  ai_addrlen;
     struct sockaddr *ai_addr;
     char * *ai_canonname;
     struct addrinfo *ai_next;
}
```

V parametru hints si lze nastavit preference nebo vynutit konkrétní protokol, rodinu protokolů a podobně (položky ai_family – může být AF_UNSPEC, dále ai_socktype a ai_protocol).

Parametr ai_flags v hints je log. součet některých z hodnot:

AI_NUMERIC – parametr node obsahuje adresu, nikoli DNS jméno.
AI_CANONNAME – dohledat do ai_canonname oficiální jméno.
AI_PASSIVE – je-li node == NULL, je ai_addr nespecifikovaná adresa. Jinak je nastaveno na zpětnovazebnou adresu.

AI_ADDRCONFIG – vrací rodinu protokolů podle toho, které protokoly jsou aktuálně na stroji konfigurované.

AI_V4MAPPED – při `hints.ai_family == AF_INET6` vrací IPv4 adresy jako IPv6 mapované, není-li žádná IPv6 adresa. Je-li navíc `AI_ALL`, vrací IPv6 a IPv4 mapované adresy.

AI_NUMERICSERV – položka `service` obsahuje číslo.

I/O operace nad sockety

recv(2), recvfrom(2), recvmsg(2) . . . Načtení dat ze socketu

```
#include <sys/types.h>
#include <sys/socket.h>
int recv(int s, void *msg, int len, unsigned flg);
int recvfrom(int s, void *msg, int len, unsigned
    flags, struct sockaddr *from, int *fromlen);
int recvmsg(int s, struct msghdr *msg,
    unsigned flags);

struct msghdr {
    caddr_t   msg_name;
    u_int     msg_namelen;
    struct    iovec *msg_iov;
    u_int     msg_iovlen;
    caddr_t   msg_control;
    u_int     msg_controllen;
    int  msg_flags;
};
```


Přečte data ze socketu. `recv(2)` se používá obvykle nad spojenými (connected) sockety. Parametr `flags` může být logický součet těchto hodnot:

MSG_OOB Zpracování Out-of-band dat.
MSG_PEEK Přečtení dat bez vymazání ze vstupní fronty.
MSG_WAITALL Načtení přesně `len` bajtů dat.

send(2), sendto(2), sendmsg(2) . . . Zaslání zprávy do socketu

```
#include <sys/types.h>
#include <sys/socket.h>
int send(int s, void *msg, int len, unsigned flg);
int sendto(int s, void *msg, int len, unsigned
    flags, struct sockaddr *to, int tolen);
int sendmsg(int s, struct msghdr *msg,
    unsigned flags);
```

Posílá data na jiný socket. `send(2)` funguje pouze na spojené (*connected*) sockety. Ostatní mohou fungovat kdykoli. Pokud je zpráva příliš dlouhá na atomický přenos, vrátí funkce chybu a `errno` nastaví na `EMSGSIZE`. Parametr `flags` může obsahovat log. součet následujících:

MSG_OOB – posílání out-of-band dat.
MSG_DONTROUTE – pouze pro přímo připojené sítě.
MSG_DONTWAIT – neblokující operace.
MSG_NOSIGNAL – nesignalizuje `SIGPIPE` v případě chyby.

◇ **Úkol:** Napište program `udpread`, který bude přijímat zprávy na zadaném UDP portu. Pro každou zprávu vypíše, odkud ji obdržel (IP adresu a port) a obsah zprávy.

◇ **Úkol:** Napište program `udpwrite`, který otevře UDP socket a bude posílat řádky standardního vstupu na daný UDP port a danou IP adresu.

Parametry socketu

getsockopt(2), setsockopt(2) Získání/nastavení parametrů socketu

```
#include <sys/types.h>
#include <sys/socket.h>
int getsockopt(int s, int level, int optname,
    void *optval, int *optlen);
int setsockopt(int s, int level, int optname,
    void *optval, int optlen);
```

Nastavení/čtení parametrů socketu. `level` je buď číslo protokolu (viz `getprotoent(3)`), nebo `SOL_SOCKET` pro úroveň socketu. Pro každý protokol existuje několik parametrů socketu, které lze nastavit. Pro úroveň socketu jsou to tyto:

SO_DEBUG – nastaví zapisování ladící informace (superuser).
SO_REUSEADDR – povolí nové použití lokální adresy při `bind(2)`.
SO_KEEPALIVE – povolí posílání keep-alive packetů.
SO_DONTROUTE – obchází směrování pro odcházející zprávy.
SO_LINGER – nastavuje chování při uzavírání socketu.
SO_BROADCAST – získání práv na posílání broadcast packetů (může pouze superuživatel).

SO_OOBINLINE – OOB data jsou čtena v normální datové frontě.
SO_SNDBUF, SO_RCVBUF – nastavení velikosti čtecího a zápisového bufferu.
SO_SNDLOWAT – low-water mark pro posílání dat.
SO_RCVLOWAT – low-water mark pro čtení dat.
SO_SNDTIMEO – timeout pro výstupní operace. Maximální doba, po kterou je proces blokovaný ve službě jádra `send(2)`.
SO_RCVTIMEO – timeout pro vstupní operace.
SO_TYPE – vrací typ socketu (například `SOCK_STREAM`).
SO_ERROR – zkoumá, došlo-li na socketu k chybě.

Viz též `socket(7)`.

shutdown(2) . . . . . . Zrušení spojení

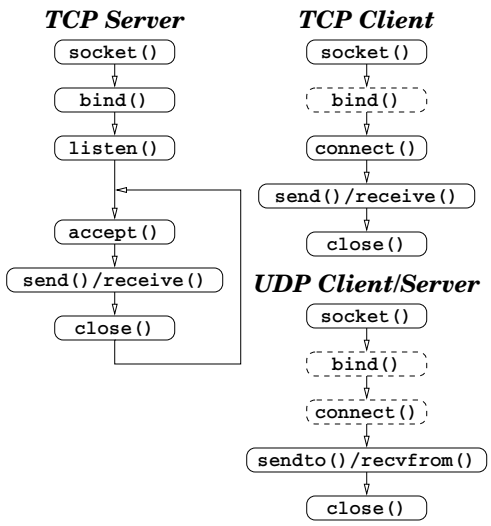
```
#include <sys/socket.h>
int shutdown(int sock, int how);
```

Zruší spojení na socketu. Parametr `how` může být jedno z následujících:

0 následující operace čtení jsou zakázány.
1 následující operace zápisu jsou zakázány.
2 všechny následující I/O operace jsou zakázány.

TCP/IP aplikace

- **Klient-server přístup.**
- **Jednoprocesový server** – Vše v jednom procesu; I/O operace multiplexovány pomocí `select(2)` nebo `poll(2)` nebo přes asynchronní I/O.
- **Víceprocesový server** – hlavní proces obvykle pouze přijme spojení přes `accept(2)`, předá potomkovi k vyřízení. Na každého klienta je jeden proces na serveru.
- **Stavový server** – výsledek předchozích operací ovlivňuje následující operace (FTP - změna adresáře).
- **Bezstavový server** – nezáleží na zopakování požadavku (NFS).



Konfigurace sítě

- Přidělení jména stroje.
- Přidělení IP adresy na interface.
- Směrovací tabulky.
- Statické versus dynamické směrování.

`hostname(1)` . . . . . Jméno stroje

```
# hostname ⟨jméno⟩
$ hostname
```

Nastaví/vypíše jméno stroje. Jméno může být FQDN (BSD) nebo jen jméno bez domény (SysV).

`uname(1)` . . . . . Jméno systému

```
$ uname [-snrvma]
```

Zjistí jméno operačního systému. Volby jsou následující:

`-m` – machine (hardware) type.
`-n` – node name (host name).
`-r` – operating system release.
`-s` – operating system name.
`-v` – operating system version.
`-a` – all of the above.

`ifconfig(8)` . . . . . Nastavení síťového rozhraní

```
$ ifconfig [⟨interface⟩]
# ifconfig ⟨interface⟩ [⟨afname⟩] ⟨options...⟩
```

Slouží k přidělení adresy síťovému rozhraní a k nastavení dalších parametrů. ⟨*afname*⟩ značí rodinu protokolů, pro kterou se provádí nastavení (`inet`, `ax25`, `ipx` a podobně). ⟨*options*⟩ je jedno nebo více z následujících (pro Linux):

`up` – aktivace rozhraní. Implicitně pokud se specifikuje nová adresa.
`down` – deaktivace rozhraní.
`address` ⟨*adresa*⟩ nebo jen ⟨*adresa*⟩ – nastavuje adresu rozhraní.
[-]**`arp`** – zapíná/vypíná použití ARP nad daným rozhraním.
[-]**`allmulti`** – rozhraní přijímá všechny multicast packety na dané síti.
[-]**`promisc`** – přijímá všechny packety na dané síti.
`metric` ⟨*N*⟩ – metrika rozhraní. Používá se pro směrovací protokoly.

`mtu` ⟨*N*⟩ – Maximum Transfer Unit.
`dstaddr` ⟨*adresa*⟩ nebo **`pointopoint`** ⟨*adresa*⟩ – cílová adresa pro point-to-point rozhraní.
`netmask` ⟨*netmask*⟩ – síťová maska rozhraní.
[-]**`broadcast`** [⟨*adresa*⟩] – nastavuje adresu pro všesměrové vysílání.
`hw` ⟨*hw-adresa*⟩ – nastavení hardwarové adresy, pokud to daný interface podporuje.
`multicast` – nastavuje multicast flag pro dané rozhraní (implicitně je nastaven na zařízeních, které toto podporují).

◇ **Příklad:**

```
# ifconfig lo 127.0.0.1 \
    netmask 255.0.0.0
# ifconfig eth0 147.251.50.60 \
    netmask 255.255.255.0 \
    broadcast 147.251.50.255
# ifconfig ppp0 down
```


Směrovací tabulka

- **Adresování na základě sítě** – adresa sítě, maska (adresní prefix).
- **Položky:** adresní prefix, adresa routeru nebo jméno rozhraní, metrika.

`route(8)` . . . . . Práce s IP směrovací tabulkou

```
$ route [-n]
```

Vypíše směrovací tabulku jádra (případně bez převodu na doménová jména).

```
# route add|del [-net|-host] ⟨target⟩ \
    [metric ⟨metric⟩] [dev ⟨interface⟩]
```

Nastaví cestu k danému počítači nebo síti. Síť `default` je totéž co `0.0.0.0/0`. Směruje se podle nejdelšího prefixu a pak podle metriky.

Program `ifconfig(8)` přidává automaticky cestu k lokálně připojené síti.

◇ **Příklad:**

```
# route add -net 127.0.0.0
# route add -net 147.251.48.0
# route add default gw 147.251.48.14
```

Směrovací protokoly

- **Dynamická modifikace směrovacích tabulek**
- **Tolerantnost k výpadku sítě**
- **Zamezení vzniku směrovacích kruhů**
- **Rozdělení zátěže**

- **Autonomní systémy** – sítě jednotlivých poskytovatelů
- **Vnitřní a vnější směrování** – v rámci nebo přes hranice AS.

- **RIP** – routing information protocol, `routed(8)`.
- **RIPv2** – odstraňuje některé nevýhody RIPu.
- **OSPF** – open shortest paths first.
- **BGP** – border gateway protocol, mezi hraničními routery autonomních systémů.

- **GateD** – `www.gated.org` – není free, značně BSD-specifický
- **Zebra** – `www.zebra.org` – GPL, portabilní, včetně IPv6.
- **BIRD** – `bird.network.cz`

Policy routing v Linuxu

- **Policy routing** – směrování podle jiného kliče, než je adresní prefix (například podle zdrojové adresy, TOS, atd.)
- **Implementace** – liší se v různých systémech.
- **Linux** – ovládání programem `ip(8)` – nahrazuje `ifconfig(8)`, `route(8)` a několik dalších.

- **Routing cache** – nemusí se pro každý datagram procházet směrovací tabulka. Indexováno podle rozhraní, TOS, zdrojové a cílové IP adresy (tedy žádné položky 4. vrstvy).
- **Směrovací tabulky** – více než jedna směrovací tabulka v jádře; předdefinované tabulky `local` a `main`.
- **Pravidla** – priorita, levá strana, pravá strana
- **Levá strana pravidla** – zdrojový prefix, cílový prefix, TOS, fwmark, zařízení.
- **Pravá strana** – číslo tabulky, nebo jedno z `prohibit`, `blackhole`, `unreachable`, dále zdrojový a cílový realm (doména).
- **Cílová doména (realm)** může být nastavena i směrovací tabulkou.

`netstat(8)` . . . . . Informace o síťovém subsystému

```
$ netstat [-vncturi]
```

Bez parametrů vypíše seznam otevřených socketů. Může mít mimo jiné následující přepínače:

- **`-v`** podrobnější výpis.
- **`-n`** číselný výpis bez převodu na doménová jména.
- **`-c`** periodický výpis podobně jako u `top(1)`.
- **`-t`** TCP sockety.
- **`-u`** UDP sockety.
- **`-r`** vypíše směrovací tabulku jádra.
- **`-i`** vypíše seznam všech síťových rozhraní.

◇ **Příklad:**

```
$ netstat -rn
Destination     Gateway         Mask Flg Iface
147.251.48.18   0.0.0.0         32   UH  eth0
147.251.48.0    0.0.0.0         24   U eth0
127.0.0.0       0.0.0.0         8    U lo
0.0.0.0         147.251.48.14 0    UG  eth0
```

`tcpdump(8)` . . . . . Výpis provozu na síti

```
# tcpdump [-exnp] [-c ⟨count⟩]
    [-i ⟨interface⟩] ⟨expression⟩
```

Poslouchá na síťovém rozhraní a vypisuje provoz na něm. Parametr ⟨*expression*⟩ říká, které packety se mají vypisovat.

- **`-n`** nepřevádí adresy na doménová jména.
- **`-x`** podrobnější výpis.
- **`-e`** vypisuje také hlavičky linkové úrovně (například MAC adresy).
- **`-p`** nepřepne rozhraní do promiskuitního režimu.
- **`-c`** ⟨*count*⟩ vypíše prvních ⟨*count*⟩ packetů.
- **`-i`** ⟨*interface*⟩ jiné než implicitní rozhraní.

◇ **Příklad:**

```
# tcpdump -i eth0 src host pyrrha \
    and dst port http
```

`ping(8)` . . . . . ICMP Echo Request

```
# ping [-nfqR] [-c ⟨count⟩] [-s ⟨size⟩] ⟨host⟩
```

Posílá ICMP echo request packety a vypisuje odezvy. Slouží k testování sítě na úrovní IP.

- **`-n`** nepřevádí IP adresy na doménová jména.
- **`-f`** flood ping.
- **`-q`** bez výstupu o jednotlivých packetech.
- **`-R`** record route.

◇ **Příklad:**

```
$ ping -c 2 pyrrha.fi.muni.cz
PING pyrrha.fi.muni.cz (147.251.50.60):
    56 data bytes
64 bytes from 147.251.50.60: icmp_seq=0
    ttl=57 time=318.2 ms
64 bytes from 147.251.50.60: icmp_seq=1
    ttl=57 time=535.2 ms
-- pyrrha.fi.muni.cz ping statistics --
2 pkts transmitted, 2 pkts received, 0% pkt loss
round-trip min/avg/max = 261.4/371.6/535.2 ms
```


IP nad Ethernetem

- **ARP** – mapování MAC adres na IP adresy.

`ifconfig up` – interface může posílat ARP reply do sítě.

`arp(8)` . . . . . Manipulace s ARP tabulkou

```
$ arp -a
# arp -d ⟨ipaddr⟩
# arp -s ⟨ipaddr⟩ ⟨hwaddr⟩ [netmask ⟨netmask⟩] \
    [pub|temp]
```

`rarp(8)` . . . . . Manipulace s RARP tabulkou

```
# rarp -d ⟨hostname ...⟩
# rarp -s ⟨hostname⟩ ⟨hwaddr⟩
```

Sériová komunikace

- **Sériový přenos** – přenos dat po jednom vodiči, bity jdou za sebou (v sérii).
- **Asynchronní přenos** – mezi počítačem a modemem: RS-232C, RS-422; mezi modemy: V.34, V.32bis. Vysílající může začít vysílat v kterémkoli okamžiku.
- **Synchronní přenos** – mezi počítačem a modemem: V.35, X.21, V.24; mezi modemy: není standardizováno. Vysílající musí začít vysílat ve stanoveném okamžiku. Předávání taktovacích informací.
- **Linková disciplína** – způsob přenosu po lince (například interpretace speciálních znaků – `BackSpace`, `Ctrl-C` a podobně). TTY vrstva v jádře UNIXu.
- **Modem** – modulátor/demodulátor – zařízení, které konvertuje datové signály za účelem přenosu po jednom nebo dvou párech vodičů, určených pro přenos zvuku.

`stty(1)` . . . . . Nastavení terminálové linky

`$ stty [-a] [-g] [`⟨*setting*⟩`]`

Bez parametrů vypíše některé informace o lince a linkové disciplíně. Slouží k nastavování parametrů.

`-a` Vypíše všechny informace o lince a linkové disciplíně.

`-g` Vypíše informace ve formátu, který může být použit jako parametr pro další `stty(1)`.

⟨*číslo*⟩ – nastavuje rychlost linky.

`cs`⟨*číslice*⟩ – počet datových bitů. Povolené hodnoty jsou 5, 6, 7 a 8.

[-]**`istrip`** – ořezávání nejvyššího bitu na vstupu.

[-]**`crtscts`** – hardware flow control.

[-]**`raw`** – neupravovaná linková disciplína.

[-]**`cooked`** – upravovaná (standardní) linková disciplína.

`intr` ⟨*znak*⟩ – přerušení (`SIGINT`) – implicitně `DEL` (`Ctrl-?`) nebo `Ctrl-C`.

`quit` ⟨*znak*⟩ – přerušení (`SIGQUIT`) – implicitně `Ctrl-\`.

`erase` ⟨*znak*⟩ – vymazání znaku – implicitně `DEL` nebo `Ctrl-H`, jinak též `BackSpace`.

`kill` ⟨*znak*⟩ – smazání řádku – implicitně `@` nebo `Ctrl-U`.

SLIP

- **Serial Line IP**
- Protokol pro přenos IP datagramů nad sériovou linkou.
- **CSLIP** – Van Jacobsonova komprese IP hlaviček.
- Minimální možnosti konfigurace.
- Doporučuje se nepoužívat.

Point-to-point protocol

- Přenos datagramů po sériových linkách.
- Protokol je symetrický – není role master a slave (neexistuje PPP server a klient).
- Práce i nad linkami bez 8-bitového přenosu.
- Posílání IP a IPX packety.
- Komprese packetů, šifrování a další transformace.
- Van Jacobsonova komprese IP hlaviček.
- Většina vlastností protokolu (escape znaky, atd.) je vyjednána při začátku spojení.
- Link Control Protocol (LCP) – vyjednávání o parametrech spojení.
- IP Control Protocol (IPCP) – vyjednávání o IP spojení.
- Autentizace – PAP, CHAP.

Synchronní přenos

- Base-band modemy.
- Linkové disciplíny: HDLC framing, nad tím synchronní PPP nebo Cisco HDLC.Frame relay.
- Komunikace nad pevnými linkami.

Cisco HDLC

- **Synchronní protokol**
- **Framing** – HDLC (linefill, kódování 5 z 6, křídlové značky).
- **Keepalive packety**
- **Dotaz na adresu** – varianta ARP.

Další přenosové protokoly

- **Synchronní PPP** – PPP zapouzdřené a kódované v HDLC rámcích.
- **PPP over Ethernet** – PPP zapouzdřené v ethernetových rámcích, modem se chová podobně jako ethernetový bridge.

Jména a adresy

- IP adresy – těžko zapamatovatelné.
- Jména počítačů.
- **Dříve:** soubor `hosts.txt` distribuovaný přes InterNIC (mechanismus stále používaný – `/etc/hosts`).
- Centrální registrace jmen počítačů.

- Nutnost rozdělení pravomoci přidělování jmen.
- Hierarchický systém jmen a adres.
- Systém domén, tečková notace *doménové adresy* (`pyrrha.fi.muni.cz`), FQDN.
- **FQDN** – řetězec jmen oddělených tečkami. Jméno může obsahovat písmena (`a-z` ze sedmibitové ASCII abecedy bez rozlišení velikosti), číslice a pomlčky (`-`). Nic jiného (RFC 1034).
- **Doména** – samostatný jmenný prostor.
- **Subdoména** – možnost vytvoření dalšího jmenného prostoru uvnitř své domény.
- **Delegace autority** spravovat subdoménu – může pouze správce nadřazené domény.
- **Kořenová doména** – značí se tečkou (`.`).

Domain Name System

- **DNS** – RFC 1033–1035, dále RFC 1912 – Common DNS Operational and Configuration Errors.
- **BIND** – referenční implementace DNS (démon `named(8)`)
- **Bližší informace** – Name Server Operations Guide for BIND (přímo v distribuci BINDu). Dále „DNS and BIND“.

- Hierarchický systém *nameserverů* – počítačů, odpovědných za jednotlivé domény.
- **Kořenové nameservery** – nameservery pro doménu „`.`“ – v Internetu mají doménové jméno `ROOT-SERVERS.NET.`, jejich seznam je zveřejňován spolu s jejich IP adresami na `ftp.internic.net`.
- **Registrovaný nameserver** – na něj jsou delegována práva spravovat subdoménu.
- **Neregistrovaný nameserver** – není odkaz z nadřazené domény, přesto zná informace o své dobméně.
- Každá doména druhé úrovně by měla mít aspoň dva registrované nameservery na navzájem různých sítích.

- **Primární nameserver** pro určitou doménu (zónu) – informace má uloženy obvykle v souboru.
- **Sekundární nameserver** pro určitou doménu – data o této zóně si stahuje periodicky z primárního nebo nadřazeného sekundárního nameserveru této zóny.
- **Cache-only nameserver** – není primární ani sekundární pro žádnou doménu, slouží pouze k přeposílání dotazů.
- **Interní kořenový nameserver** – kořenový nameserver v síti s protokolem TCP/IP, která není připojena do Internetu.
- Nameserver ví, koho se ptát, když on sám neví (například zná kořenové nameservery).
- **Reverzní mapování** (převod IP adres na jména – pseudodoména `IN-ADDR.ARPA`). Přidělení bloku IP adres k používání s sebou nese povinnost udržovat příslušný primární nameserver pro danou doménu `IN-ADDR.ARPA`.
- **Rekurzivní chování** – DNS server přeposílá dotazy sám.
- **Iterativní chování** – DNS server vrací „nevím, ale zeptejte se támhle“.

Konfigurace BINDu

`named(8)` . . . . . Domain Name Server

```
# named [-d ⟨debuglevel⟩] [-p ⟨port⟩[/⟨localport⟩]] \
    [-c ⟨configfile⟩] [-u ⟨user⟩] [-g ⟨group⟩] [-t ⟨rootdir⟩]
```

Spouští domain nameserver. Standardní konfigurační soubor je `/etc/named.conf`. Spouštět jako nesuperuživatelský proces (`-u` a `-g`) v chrootovaném prostředí (`-t`).
Program reaguje na následující signály:

`SIGHUP` – způsobí znovunačtení konfigurace
`SIGINT` – uloží aktuální databázi a cache do souboru `/var/tmp/named_dump.db`.
`SIGUSR1` – Zvýší ⟨*debuglevel*⟩ o jedničku.
`SIGUSR2` – Vypne ladění (⟨*debuglevel*⟩ 0).

Soubor `named.conf`

```
options {
        directory "/var/named";
        forwarders {
                147.251.48.3; 147.251.4.33;
                3ffe:ffff::1;
        };
        forward only;
};
zone "." {
        type hint;
        file "named.root";
};
zone "localhost" {
        type master;
        file "master/localhost";
        allow-update { none; };
        allow-transfer { any; };
};
```



```
zone "fi.muni.cz" {
        type master;
        file "master/fi.muni.cz";
        also-notify {
                147.251.48.140;
                3ffe:ffff::1;
        };
        allow-transfer { any; };
};
zone "muni.cz" {
        type slave;
        file "slave/muni.cz";
        masters { 147.251.4.33; };
        allow-transfer { any; };
};
zone "0.0.127.in-addr.arpa" {
        type master;
        file "master/127.0.0";
        allow-update { none; };
};
```



```
zone "0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.\
      0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.arpa" {
        type master;
        file "master/0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.\
             0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0";
        allow-update { none; };
        allow-transfer { any; };
};
zone "48.251.147.in-addr.arpa" {
        type master;
        file "master/147.251.48";
        also-notify { 147.251.48.140; };
};
zone "f.f.f.f.e.f.f.3.ip6.arpa" {
        type master;
        file "master/3.f.f.e.f.f.f.f";
        also-notify { 3ffe:ffff::1; };
};
```


Dopředný překlad

Příklad souboru `fi.muni.cz`:

```
$TTL 1D
@          IN SOA  anxur.fi.muni.cz. \
                   postmaster.fi.muni.cz. (
                   2001041701  ; Serial
                   3H          ; Refresh
                   15M         ; Retry
                   2W          ; Expire
                   1D    )     ; Minimum TTL

           IN NS   anxur
           IN NS   aisa
           IN NS   ns.ten.cz.
           IN MX   50 relay.muni.cz.
           IN MX   100 relay
           IN TXT  "Faculty of Informatics"
           IN TXT  "Masaryk University Brno"
```

```
ns          IN CNAME anxur
ftp         IN CNAME odysseus
news        IN CNAME nimloth.ics.muni.cz.
time        IN CNAME arkas
relay       IN A 147.251.48.3
anxur       IN A 147.251.48.3
;           IN HINFO        "SUN SS10" "Solaris 2"
;           IN TXT          "Umisteni: M.Brandejs"
calypso     IN A 147.251.50.61
            IN AAAA 3ffe:ffff::3d
;           IN HINFO        "PC" "Linux"
;           IN TXT          "Umisteni: Jan Kasprzak"
            IN MX   50 relay.muni.cz.
            IN MX   100 relay
...
```


Root cache

- Seznam kořenových nameserverů.
- Není nutné u *forward-only* nameserverů.

Příklad souboru `named.root`:

```
; formerly NS.INTERNIC.NET
.                3600000 IN NS A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET. 3600000 IN A 198.41.0.4
; formerly NS1.ISI.EDU
.                3600000 IN NS B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET. 3600000 IN A 128.9.0.107
; formerly C.PSI.NET
.                3600000 IN NS C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET. 3600000 IN A 192.33.4.12
; formerly TERP.UMD.EDU
.                3600000 IN NS D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET. 3600000 IN A 128.8.10.90
; formerly NS.NASA.GOV
.                3600000 IN NS E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET. 3600000 IN A 192.203.230.10
...
```


Reverzní mapování

Příklad souboru `147.251.48`:

```
$TTL 1D
48.251.147.in-addr.arpa. IN SOA anxur.fi.muni.cz.\
                         postmaster.fi.muni.cz. (
                         2001041700 ; Serial
                         3H         ; Refresh
                         15m        ; Retry
                         2W         ; Expire
                         1D    )    ; Minimum

                  IN NS   anxur.fi.muni.cz.
                  IN NS   aisa.fi.muni.cz.

1                 IN PTR  aisa.fi.muni.cz.
3                 IN PTR  anxur.fi.muni.cz.
                  IN PTR  relay.fi.muni.cz.
5                 IN PTR  thetis.fi.muni.cz.
...
```


Příklad souboru `3.f.f.e.f.f.f.f`:

```
$TTL 1D
f.f.f.f.e.f.f.3.ip6.arpa. IN SOA \
                         anxur.fi.muni.cz.\
                         postmaster.fi.muni.cz. (
                         2001041700 ; Serial
                         3H         ; Refresh
                         15m        ; Retry
                         2W         ; Expire
                         1D    )    ; Minimum

                  IN NS   anxur.fi.muni.cz.
                  IN NS   aisa.fi.muni.cz.

d.3.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0 \
                  IN PTR  calypso.fi.muni.cz.
...
```


Resolver

- Mechanismus pro překlad doménových adres a IP adres.
- Funkce `gethostbyname(3)` a `gethostbyaddr(3)`.
- Další funkce, související s DNS: `res_query(3)`, `res_search(3)`, `res_send(3)`, `res_mkquery(3)`, `res_init(3)`.
- Možnost přepínání mezi různými mechanismy resolvování adres.

`host.conf(5)` . . . . . Konfigurace resolveru na Linuxu

```
order hosts,bind
multi on
```

- Nastavuje pořadí resolvovacích mechanismů.
- **Solaris:** Podobný soubor `nisswitch.conf(5)`.
- **IRIX:** Řádek typu „`hostresorer local bind`“ v souboru `resolv.conf(5)`.

`resolv.conf(5)` . . . . . Nastavení nameserverů a domény

```
domain fi.muni.cz        nebo:
search fi.muni.cz ics.muni.cz muni.cz
nameserver 147.251.48.140
nameserver 147.251.50.1
nameserver 147.251.4.33
```

- Lze nastavit jméno domény, vzhledem ke které se provádí DNS dotazy.
- *search list* – nastavení dalších domén pro relativní dotazy.
- Seznam nejvýše čtyř nameserverů (má vliv na velikost timeoutu).

Další služby sítě

- **TCP servery** – podobný mechanismus činnosti.
- Smyčka `socket()`, `bind()`, `listen()`, `accept()`.
- Jednodušší řešení: *servlet*, který očekává již spojený socket na standardním vstupu/výstupu + program, který bude čekat na více portech a akceptovat spojení.

Internet super-server

- **Program `inetd(8)`**
- **Výhoda** – menší kód TCP a UDP serverů, nižší nároky na systémové prostředky (nemusí být pro každou službu aktivní alespoň jeden proces).
- **Konfigurační soubor** `/etc/inetd.conf`

Alternativy `inetd`

`xinetd` – větší možnosti konfigurace, omezení přístupu, nastavení priorit, atd.

`tcpserver` – samostatný program na spouštění servletů. Pro každou službu musí běžet jeden.

Příklad `inetd.conf`

```
# <service_name> <sock_type> <proto> \
        <flags> <user> <server_path> <args>
echo    stream tcp  nowait root internal
echo    dgram  udp  wait   root internal
discard stream tcp  nowait root internal
discard dgram  udp  wait   root internal
daytime stream tcp  nowait root internal
daytime dgram  udp  wait   root internal
chargen stream tcp  nowait root internal
chargen dgram  udp  wait   root internal
ftp     stream tcp  nowait root \
        /usr/sbin/in.ftpd in.ftpd -la
telnet  stream tcp  nowait root \
        /usr/sbin/in.telnetd in.telnetd
```


TCP wrapper

- **Bezpečnost** – omezení použití TCP služeb podle IP adres/domén.
- **Firewall/packetový filtr** – nákladné nebo špatně dostupné řešení pro některé systémy, nutnost věřit lokální síti.
- **Aktivní bezpečnost** – nejen ověřování příchozích spojení, ale i možnost aktivní obrany.
- **Použití** – jako knihovna `libwrap.a` – `hosts_access(3)`, wrapper pro servlety z `inetd(8)` – program `tcpd(8)`. `/usr/sbin/tcpd in.telnetd`
- **Konfigurace** – soubory `hosts.allow` a `hosts.deny`, manuálová stránka `hosts_access(5)` a `hosts_options(5)`.
- **Dvojitý DNS dotaz.**

Syntaxe `hosts.allow`, `hosts.deny`

- Řádky.
- Prázdný řádek a řádek, začínající křížkem se ignoruje.
- Zbytek má syntaxi:
  ⟨*daemon-list*⟩ **:** ⟨*client-list*⟩ [**:** ⟨*shell-command*⟩]
  ⟨*daemon-list*⟩ **:** ⟨*client-list*⟩ [**:** ⟨*options*⟩] ...

◇ **Příklad:**

```
in.telnetd : .evil.domain : echo "denied telnet \
     from user %c" | mail -s"security attack" root
in.ftpd : .evil.domain 10.0.
```

je-li `-DPROCESS_OPTIONS`:

```
in.telnetd : .evil.domain : spawn \
     (/some/where/safe_finger -l @%h \
     | /usr/ucb/mail root) &
```


Protokol ident

- **RFC 931, RFC 1413**
- **Zjištění jména uživatele k existujícímu TCP spojení**
- **Pro audit, nikoliv pro autentizaci**
- **Na dotaz se nesmí reagovat jinak než odpovědí!**
- **Ident-démon** – samostatně běžící, nebo z `inetd`.
- **Šifrování** – zamezení úniku informací – neposílá se jméno uživatele, ale zašifrovaná data. Rozšifrovat může pouze administrátor původního serveru.
- **TCP wrapper** – podpora ident-dotazů: místo IP adresy se uvede ⟨*user*⟩`@`⟨*adresa*⟩:

```
ALL : ALL@ALL
ALL : ALL : rfc931
```

```
Apr 29 19:10:44 erinys10 fingerd[18038]: \
     refused connect from \
     mikulas@artax.karlin.mff.cuni.cz
```


Telnet

- Emulace terminálové linkové disciplíny.
- Server – démon `telnetd(8)` – má podobnou funkci jako `getty(8)` pro sériové linky.
- Pseudoterminál – zařízení podobné rouře, nad kterým je možno použít terminálovou linkovou disciplínu.
- Autentizace: jméno/heslo – pozor, jde v síti nezašifrovaně. Existuje `telnet` s autentizací Kerberos.
- Telnet klient: Pod UN*Xem běžný program, pod jinými systémy má v sobě obvykle emulátor terminálu.

`telnet(1)` . . . . . Vzdálené přihlášení

`$ telnet` [`-d`] [`-e` ⟨*znak*⟩] [⟨*host*⟩ [⟨*port*⟩]]

Program je podobný programu `cu(1)` pro sériové linky. Naváže spojení na daný počítač protokolem *telnet*, případně přímo na určitý port. Nezadáme-li počítač, dostaneme prompt telnetu a můžeme zadávat přikazy.

`-d` zapíná ladící výpisy protokolu.

`-e` nastaví escape znak pro přepnutí do příkazového režimu (escape znak pošleme na druhý konec pomocí `send escape`). Implicitní escape znak je `^]`.

r-služby

- `rlogin(1)`, `rcp(1)` a `rsh(1)`.
- Autentizace – `rlogin`: jméno a heslo nebo `rhosts`, ostatní služby jen `rhosts`.

rhosts – důvěřuje druhé straně co se týče jména uživatele, proto musí spojení přicházet z privilegovaného portu.

hosts.equiv – které počítače jsou považovány za ekvivalentní z hlediska oprávněných uživatelů.

---

rlogin(1) . . . . . Vzdálené přihlášení

`$ rlogin [-8] [-E ⟨znak⟩] [-l ⟨username⟩] ⟨host⟩`

-8 – nastaví 8-bitový přenos.

-E – nastaví escape znak (implicitně tilda, na druhý konec se pošle zopakováním znaku).

Secure Shell

- `telnet` – heslo jde po síti v otevřené podobě; možnost odposlechu hesla.
- `rlogin` – ověřování na základě IP adresy; důvěřuje vzdálenému počítači při zjišťování jména uživatele.
- Možnost IP spoofingu (přesměrování IP adresy a podobně).
- Možnost DNS spoofingu.
- Obě služby – možnost *man in the middle attack*.

Principy činnosti SSH

- Komunikace mezi nad nedůvěryhodnou sítí.
- Ověřování totožnosti klienta, ale i serveru.
- **Server** – program `sshd(8)`.
- **Klient** – program `ssh(1)`. Funkce obdobná jako u `rlogin(1)` a `rsh(1)`.

SSH verze 1

- **Host key** – každý SSH server vlastní svůj pár RSA klíčů. Slouží k ověřování pravosti stroje a znemožňení man in the middle útoků.
- **Server key** – `sshd(8)` má další pár RSA klíčů, který vznikne při startu programu a který se čas od času (např. po hodině) mění.
- **Server** pošle klientovi veřejný klíč stroje a serveru. Dále server posíla seznam šifer, které podporuje.
- **Klient** ověří, jestli se nezměnil veřejný klíč stroje.
- **Klient** vygeneruje 256-bitové náhodné číslo, zašifruje je pomocí obou klíčů, vybere šifru a vše pošle je serveru.
- **Další spojení** – zašifrováno konvenčním agloritmem (DES, IDEA, 3DES, ARCFOUR) tímto až 256-bitovým klíčem.

SSH verze 2

- **Nový protokol** – nekompatibilní s verzí 1.
- **Host key** – může být RSA nebo DSA.
- **Server key** – zde není. Dohoda nad klíčem pro symetrickou šifru algoritmem Diffie-Hellmann.
- **Zabezpečení integrity** – HMAC-MD5 nebo HMAC-SHA1.

Ověřování totožnosti klienta

- Různé způsoby, každý z nich lze zapnout nebo vypnout.
- **`.rhosts` autentizace** – stejná jako u r-služeb. Není bezpečné. Implicitně zakázané. V SSH2 není.
- **`.rhosts` + RSA ověření stroje** – podobné jako předchozí, jen jméno vzdáleného stroje se navíc ověří tak, že stroj musí prokázat svoji totožnost svým tajným RSA klíčem (obrana proti chosen-plaintext útoku na RSA).
- **Ověření heslem** – podobné jako u `telnet(1)`. Heslo jde po síti v zašifrované podobě, neboť celé spojení je v této fázi zašifrované.
- **RSA autentizace** – ověření pravosti uživatele pomocí jeho RSA klíče (možnost spolupráce s čipovou kartou). V SSH2 možno i jiné typy asymetrických klíčů (DSA).
- **Další možnosti** – obecný protokol challenge-response (jen v SSH2).

Další vlastnosti

- **Po ověření přístupových práv** – klient zašle příkaz, který se má vykonat a další parametry (alokace pseudoterminálu a podobně).
- **Zvláštní kanál pro std. výstup a chybový výstup** – `rsh(1)` nemá.
- **Forwardování portů** – zajistí šifrované TCP spojení mezi dvěma porty na dvou strojích (dobré pro použití dalších služeb, u kterých jde heslo po síti v otevřené podobě).
- **Forwardování X11 spojení** – výměna `.Xauthority`, další „lokální“ X server, forwardování akcí na vzdálený server šifrovaným kanálem.
- **SSH agent** – funguje podobně jako čipová karta. Uživatel nemusí znovu zadávat passphrase svého asymetrického klíče.
- **Agent forwarding** – `ssh(1)`, který je vyvolaný ze vzdáleného spojení, může mít přístup ke klíčům lokálního agenta.

Nevýhody

- **Snížení propustnosti** – šifrování je pomalé (X11, remote backup).
- **Nemožnost rozumného použití** u X-terminálu nebo NC.
- **Existence `sshd(8)` v systému** – server nelze spouštět přes `inetd(8)` z důvodu nutnosti generování RSA páru klíčů při každém startu.

File Transfer Protocol

- **Protokol pro přenos souborů**
- **Možnost přenosu i mezi dvěma vzdálenými počítači**
- **Řídící spojení** – `21/tcp`, iniciováno klientem.
- **Datové spojení** – zdrojový port (obvykle) `20/tcp`, iniciováno serverem.
- **Pasivní FTP** – i datové spojení otevírá klient na server. Někdy lepší průchod přes proxy-servery a firewally.
- **Anonymní FTP** – obvykle přihlášení na dohodnutý účet `ftp` nebo `anonymous`. Povoluje se pouze čtení určitého podstromu (a FTP-démon pro tento strom zavolá `chroot(2)`).

FTP servery

- **WU-FTPD** – Washington University, nejrozšířenější. Podporuje anonymní i neanonymní FTP, třídy uživatelů, atd.
- **ProFTPD** – virtuální FTP servery, konfigurace podobná HTTP serveru Apache, modularita, IPv6.
- **Troll Tech FTPD** – minimalistický, pouze pro anonymní FTP, vestavěný program `ls`.
- **Ostatní** – PureFTPd, NcFTPd, vsFTPd, atd.

Remote Procedure Call

- **Vzdálené volání procedury**
- Vykonání procedury asynchronně na vzdáleném stroji.
- Sun RPC a TIRPC, OSF DCE.
- Jednoznačné číslo procedury/služby, verze protokolu.
- Formát dat nezávislý na platformě.
- Sun XDR – external data representation. Použitelné pro RPC nebo například pro ukládání dat do souboru a výměnu dat mezi platformami.

`rpc.portmap(8)` . . . . . Port mapper

- RPC služeb může být mnoho, proto není jednoznačná korespondence mezi RPC službou a TCP nebo UDP portem.
- Porty pro RPC služby alokovány dynamicky, mapování čísla služby a verze na port zajišťuje portmapper.
- Forwardování požadavků.
- Vzdálená registrace služeb.
- **Secure portmapper** – některé bezpečnostní kontroly, knihovna TCP wrapperu. Autorem je Wietse Venema.

`rpcinfo(8)` . . . . . Informace od portmapperu

```
$ rpcinfo -p
   program vers proto   port
    100000    2   tcp    111  rpcbind
    100000    2   udp    111  rpcbind
    100005    1   udp    766  mountd
    100005    1   tcp    768  mountd
    100005    2   udp    771  mountd
    100005    2   tcp    773  mountd
    100003    2   udp   2049  nfs
```


Network File System

- Sdílení souborů po síti.
- RPC služba.
- Bezestavová služba – výhody i problémy.
- Není autentizace – sdílený prostor UID/GID nebo `rpc.ugidd`.

Klient

```
# mount -t nfs aisa:/export/home /home
```

- Přístup je podobný jako k lokálním diskům.
- Nemožnost zamykání existencí souboru.
- Zamykání části souboru – démon `rpc.lockd(8)` a `rpc.statd(8)` bez veřejně dostupné specifikace.
- Asynchronní zápis/čtení – démoni `biod(8)` nebo `nfsiod(8)` uvnitř jádra.

Server

- **Démoni** `rpc.mountd(8)`, `rpc.nfsd(8)`, případně `rpc.ugidd(8)` pro mapování UID a GID.
- Někdy je `nfsd(8)` uvnitř jádra a běží v několika instancích pro urychlení konkurentního přístupu.
- `nfsd(8)` má pevně vyhrazený port 2049, přestože jde o RPC službu.
- Seznam adresářů, které jsou přístupné zvenku: `/etc/exports`, viz `exports(5)`.
- `showmount(8)` – výpis informací z mount-démona (export-list, seznam připojených adresářů).

Network Information System

- **Dříve YP** – Yellow Pages; nyní NIS, NIS+.
- **Sdílení systémových tabulek po síti** – napojeno na nsswitch.
- **Server** – démon `ypserv(8)`.
- **Primární NIS/YP server, záložní NIS/YP servery**
- **Klient** – démon `ypbind(8)`.
- **Sdílení `/etc/passwd`** – démon `yppasswdd(8)` pro změnu hesla.
- **Binární podoba tabulek** – `ypmake(8)`.

Kerberos

- **Centrální autentizační systém**
- **Třístranná autentizace** – uživatel, služba/server, KDC.
- **Key Distribution Center** – služba, která má databázi všech hesel.
- **Lístky** (tickets) – vydává KDC, používají se pro prokazování totožnosti klienta vůči službě.
- **Heslo** (tajemství/secret) – pro uživatele i pro služby.
- **Realm** – doména; oblast spravovaná jedním KDC.
- **Principal** – jméno uživatele nebo služby (fyzická osoba takto může vystupovat pod více identitami).

Literatura

- **Designing an Authentication System** (rozhovor ve čtyřech scénách) – *http://web.mit.edu/kerberos/www/dialogue.html*.
- **Kerberos v5 Installation Guide** – kompilace systému, základní administrativní rozhodnutí.
- **Kerberos v5 Administrator's Guide** – konfigurační soubory, administrace databáze uživatelů.
- **Kerberos v5 User's Guide** – základní informace pro uživatele.

Získání/použití lístku

- **Žádost o lístek** – jméno (principal) uživatele, adresa, jméno (principal) služby.
- **Odpověď** – lístek, (session key, jméno klienta, jméno serveru) zašifrováno heslem uživatele.
- **Uživatel** – zadáním hesla získá session key.
- **Obsah lístku** – (session key, jméno a adresa klienta, jméno serveru, čas, doba platnosti) zašifrováno heslem služby.
- **Autentizace** – klient vytvoří autentizátor a pošle i s lístkem službě.
- **Autentizátor** – (jméno a adresa klienta) zašifrováno session key.
- **Ověření** – služba rozšifruje lístek, získá session key, rozšifruje autentizátor a ověří, že si jméno a adresa klienta odpovídá.

Další vlastnosti

- **Ticket granting service** je také služba.
- **Ticket granting ticket** – TGT – lístek pro získávání dalších lístků.
- **Získání TGT** – program `kinit(1)`. Dále `klist(1)`, `kdestroy(1)`.
- **Replay cache** – proti odposlechnutí.
- **Autentizace služby vůči klientovi.**
- **Forwardovatelné lístky** – nejsou vázané na adresu.
- **Proxy lístky** – delegace části pravomocí na službu.
- **Interakce mezi realmy** – například cross-realm trust.
- **Záložní KDC** – replikace.

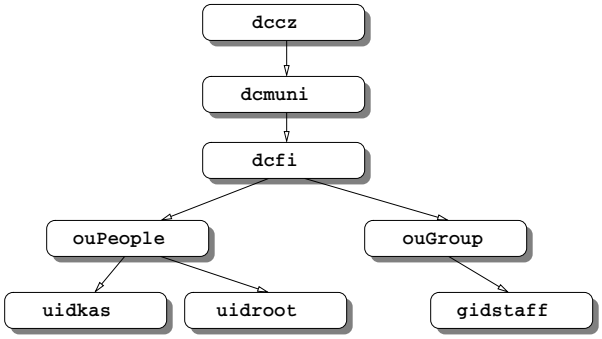
Implementace

- **MIT Kerberos V** – referenční implementace.a
- **Heimdal** – evropská implementace.
- **Klientské aplikace** – nutnost kerberizace (mj. rozšíření protokolu).
- **Konfigurace klientů** – `/etc/krb5.conf`.

LDAP

- **Light-weight Directory Access Protocol**
- **Původ** – odlehčená varianta DAP pro přístup k X.500 adresářům.
- **Adresář** – ne jako ve filesystému. Analogie telefonního seznamu.
- **Adresářová služba** – databáze pro rychlé vyhledávání a občasné modifikace dat.

LDAP strom



LDAP – vlastnosti

- **Celosvětový distribuovaný adresář** – i odkazy mezi LDAP servery.
- **Uzly stromu – objekty**
- **Distinguished name** – DN – poloha ve stromu (cesta ke kořeni) – např.: `uid=kas,ou=People,dc=fi,dc=muni,dc=cz`
- **Relative DN** – v rámci jedné úrovně (`uid=kas`).
- **Objekty** – patří do tříd (i více), mají různé atributy.
- **Schéma** – definice tříd (názvy atributů, typy hodnot, povinné, nepovinné atributy). Definice ve formátu ASN.1.
- **Vyhledávání** – v daném podstromu, objekt určitých atributů.
- **Odkazy** – referrals – něco jako symbolické linky.
- **Read-mostly databáze** – nejsou transakce, rychlé vyhledávání.

Autentizace v LDAPu

- **Autentizace proti LDAP serveru** – externí (SASL), simple (heslo je součástí objektu). Anonymní přístup.
- **Subjekt práv** – obecné distinguished name. Není zvlášť databáze uživatelů.
- **Objekt práv** – podstrom, jednotlivá DN, *self*, jednotlivé atributy.
- **Přístupová práva** – *authenticate*, *compare*, *read*, *search*, *write*.

Formát LDIF

- **Textový formát pro výměnu dat mezi servery**

```
$ ldapsearch -H ldap://ldap.fi.muni.cz/ \
   -b ou=People,dc=fi,dc=muni,dc=cz uid=kas -x

# kas, People, fi.muni.cz
dn: uid=kas,ou=People,dc=fi,dc=muni,dc=cz
uid: kas
cn: kas
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
userPassword:: e2NyeXB0fXg=
loginShell: /bin/bash
uidNumber: 11561
gidNumber: 10100
homeDirectory: /home/kas
gecos: Jan Kasprzak
host: aisa
host: anxur
```


Použití v UNIXu

- **Napojení na nsswitch** – možnost uložení tabulek uživatelů, skupin, služeb, protokolů ...
- **Modul `nss_ldap`** – konfigurace v `/etc/ldap.conf`. Mapování podstromu LDAP objektů a jejich atributů na záznamy v tabulce uživatelů.
- **Automatický převod** – MigrationTools – `www.padl.com`.
- **Uživatelé** – třídy `account`, `posixAccount` a `shadowAccount`.
- **Zabezpečení** – možnost použít LDAP nad SSL (`nss_ldap` umí kontrolovat i certifikát serveru).

Ukládání hesel v LDAPu

- **Několik šifrovacích metod**
- **Formát hesla** – např.: `{SMD5}F92mezjPoWxSE`.
- **Hashovací metody** – `{CRYPT}`, `{SMD5}`, `{MD5}`, `{SSHA}`, `{SHA}`.
- **Generování hashované podoby hesla** – `slappasswd(8)`.
- **Heslo v LDIF formátu** – často kódované base64. Oddělovač `::` v LDIF. Podobně se kódují binární data.

Implementace LDAPu

- **Řádkoví klienti** – `ldapsearch(1)`, `ldapadd(1)`, `ldapdelete(1)`, man `ldapmodify(1)`.
- **Ostatní klienti** – `GQ(1)`, nss_ldap, Mozilla.
- **Servery** – OpenLDAP, iPlanet/SunONE, Oracle Internet Directory, Active Directory, ...

Elektronická pošta

- **Neinteraktivní (off-line) komunikace** – zprávy jsou odeslány a uloženy na cílovém počítači kde čekají na přečtení.
- **Textová komunikace** – později i možnost začlenění dalších druhů dat (zvuk, grafika, binární soubory, atd).
- **MTA** – mail transport agent – program/systém pro přenos zpráv po síti (`sendmail`, `qmail`, `exim`).
- **MUA** – mail user agent – uživatelský program pro čtení a posílání zpráv (`elm`, `exmh`, `mutt`).
- **MDA** – mail delivery agent – program spouštěný MTA při doručení do lokální schránky (`mail(1)`, `procmail(1)`, `deliver(1)`).

Obálka zprávy

- **Obálka** $\neq$ **hlavičky**
- **Doručování zprávy** – podle obálky.
- **Obálkový odesílatel** – pro chybové zprávy.
- **Obálkový příjemce** – skutečný příjemce zprávy.
- **Chybová zpráva** – prázdný odesílatel.

Formát zpráv

- **Původní formát** – RFC 822, nyní RFC 2822. „Standard for ARPA Internet Text Messages“.
- **Hlavička** – obsahuje řídící informace zprávy a má pevnou strukturu.
- řádky tvaru ⟨*klíč*⟩`:` ⟨*parametry*⟩
- řádky tvaru ⟨*bílý znak*⟩⟨*parametry*⟩ – pokračování parametrů ke klíči z předchozího řádku.
- prázdný řádek – ukončení hlaviček zprávy. Zbytek je *tělo zprávy*.
- řádky by neměly být delší než 80 znaků (nejen z důvodů čitelnosti), a není-li to v hlavičce výslovně uvedeno (`Content-Transfer-Encoding: 8bit`), nesmí obsahovat znaky jiné než bílé a 32–126.

Původce zprávy

`From:` mailbox autora/autorů zprávy.
`Sender:` skutečný odesílatel zprávy, je-li ve `From:` více mailboxů.
`Reply-To:` – na jakou adresu se má poslat odpověď.

Adresát zprávy

`To:` adresa hlavního příjemce.
`Cc:` – carbon copy – další příjemci.
`Bcc:` – blind carbon copy – totéž, ale tato hlavička se při odesílání ze zprávy odstraní.

Identifikace

`Message-Id:` – jednoznačná identifikace zprávy. Používá se například k detekci zacyklení pošty.
`In-Reply-To:` – „v odpovědi na“ – identifikátor předchozí zprávy (na kterou tato zpráva odpovídá).
`References:` – identifikátory předchozích zpráv.

Trasování zprávy

`Received:` – každý MTA po cestě přidá jeden takovýto řádek se služebními informacemi.
`Return-Path:` – u doručené zprávy cesta k odesílateli.

Ostatní položky

`Date:` – datum vzniku zprávy.
`Subject:` – předmět, věc.
`Keywords:` – klíčová slova.
`X-`⟨*cokoli*⟩ – nestandardní hlavičky (`X-Face:`).

MIME

- **MIME** – Multipurpose Internet Mail Extensions. Rozšíření RFC 822, popsané v RFC 1521, RFC 2045–2049 a RFC 2231.
- **Povinné hlavičky**
  `Mime-Version: 1.0`
  `Content-Type:` ⟨*typ*⟩[⟨*podtyp*⟩[`;` ⟨*parametry*⟩ `...`]]
  `Content-Transfer-Encoding:` ⟨*přenosové kódování*⟩
- **Typy zpráv** – registruje IANA.

Jednoduché typy/podtypy:

`text` / `plain`, `html`, `richtext`, ...
`image` / `gif`, `jpeg`, `g3fax`, ...
`audio` / `basic`, ...
`video` / `mpeg`, `quicktime`, ...
`application` / `octet-stream`, `postscript`, `pdf`, ...
`x-`*nestandardní*

Složené typy/podtypy

`multipart/mixed`
– více objektů různých typů.
`multipart/parallel`
– paralelně prezentovatelné části (například text a zvuk).
`multipart/alternative`
– MUA má zobrazit jednu z částí (například `text/plain` a `text/html`).

Příklad hlaviček

```
Return-path: fikera@informatics.muni.cz
Received: from anxur.fi.muni.cz
   (11876@anxur.fi.muni.cz [147.251.48.3])
   by aisa.fi.muni.cz (8.8.5/8.8.5) with ESMTP id
   PAA08513; Thu, 21 May 1998 15:15:02 +0200 (MET DST)
Received: (from fikera@localhost)
   by anxur.fi.muni.cz (8.8.5/8.8.5) id PAA12399
   for unix; Thu, 21 May 1998 15:14:57 +0200 (MET DST)
Message-id: <199805211314.PAA12399@anxur.fi.muni.cz>
X-mailer: ELM [version 2.4ME+ PL39 (25)]
Mime-version: 1.0
Content-type: text/plain; charset=US-ASCII
Content-transfer-encoding: 7bit
From: Marek Fikera <fikera@informatics.muni.cz>
```

Formát mailboxů

- **Mailbox** – místo, do kterého se ukládají zprávy.
- **`mbox`** – standardní formát pod UN*Xem. Zprávy jsou uloženy za sebou v souboru, zpráva začíná řetězcem `From`⟨*mezera*⟩ na začátku řádku. Následuje adresa odesílatele a datum přijetí. Na dalším řádku pak následuje vlastní zpráva. Řádky, začínající řetězcem „`From ` “ jsou odsazeny jiným znakem.
- **MMDF folder** – podobné jako mailbox, ale zprávy jsou odděleny čtyřmi znaky `Ctrl-A`.
- **MH-folder** – adresář; jednotlivé zprávy jsou uloženy v souborech s číselnými názvy. Soubory označené ke smazání mají název začínající čárkou.
- **Maildir** – formát Qmailu. Tři adresáře: `tmp`, `new`, `cur`. Odstraňuje problémy se zamykáním a současným přístupem z více strojů.

SMTP

- **SMTP** – Simple Mail Transfer Protocol – protokol pro přenos pošty nad TCP/IP.
- **Definice** – RFC 821, nověji RFC 2821.
- **Další rozšíření** – ESMTP (8-bitový přenos, ETRN, atd).
- **Inicializace** – `HELO`, případně `EHLO` *jméno*.
- **Předání obálky** – `MAIL FROM:`<*odesílatel*>, `RCPT TO:`<*adresát*>.
- **Předání zprávy** – `DATA`; zakončeno tečkou na samostatném řádku.
- **Ukončení relace** – `QUIT`.
- **Vynucený přenos** – `ETRN`.
- **Kontrola odesilatele** – `VRFY`.
- **Expanze aliasů** – `EXPN`.

Lokální klienti

- **Přímý přístup k mailboxu** (zamykání). Set-gid pro skupinu `mail`.
- **Odesílání** – na vstup `/usr/sbin/sendmail -t`.

Síťoví klienti

- **Čtení pošty** – POP-3 nebo IMAP.
- **Odesílání** – SMTP přes relay.

POP-3

- **Post Office Protocol**
- **Výlučný přístup k mailboxu**
- **Atomická operace během celé session**

IMAP

- **Internet Mail Access Protocol**
- **Sdílený přístup k mailboxu**
- **Práce s více schránkami v rámci jednoho účtu**
- **Možnost čtení pošty z více počítačů**

X Window System

- **Názvy** – X, X Window System, X Version 11, X Window System Version 11, X11.
- **Historie** – DEC 1983–1986. Hlavní architekt Jim Gettys.
- **X Consortium** – řídí další vývoj (nyní The Open Group).
- **XFree86** – původně implementace pro PC, nyní i jiné architektury.

Architektura

- **X server** – proces nebo zařízení, které zobrazuje okna. `XF86_SVGA`, `XSun`.
- **X klient** – aplikace, která vyžaduje zobrazování. `xterm`.
- **Spojení** – socket (nezávislé na nižší vrstvě).
- **X protokol** – odděluje server a klienta.
- **Extenze protokolu** – pro speciální případy. Mit-SHM, XKB, Shape.

X Server

- **Poslouchá** na socketech, obsluhuje klienty.
- **Implementace** – proces, X terminál.
- **Display** – X server.
- **Screen** – obrazovka.
- **Pojmenování** – `[`*stroj*`]:`*displej*`[.`*obrazovka*`]`
- **Příklad** – `:0`, `aisa:12.0`, `unix:0.0`, `alpha::0`.
- **Klient** – proměnná `DISPLAY`, přepínač `-display`.
- **Informace** – `xdpyinfo`
- **Autentizace klienta.**
- **X Klient** – obvykle se zobrazuje na jednom displeji.

Okna

- **Hierarchie** – strom (kořenové okno, podokna). Oříznutí potomků na velikost rodiče.
- **Visual** – způsob zobrazení. Z množiny nabízené X serverem.
- **Další atributy** – pozadí, barva popředí, okraj, kurzor, gravitace, maska událostí.
- **Obsah okna** – nemusí být uchováván (klient může uchování doporučit).
- **Saveunder** – pro pop-up menu.

Kurzory

- **Okno** – svůj typ kurzoru.
- **Bitmapy** – tvar a vzhled.
- **Kurzorový font** – předdefinované kurzory.
- **Focus** – on-click, on-move.

Barvy

- **RGB**
- **Pojmenování** – `showrgb`.
- **Konverze** mezi barevnými prostory.
- **Barevné palety** (colormap) – každé okno může mít svoji, přepne se při fokusu.

Fonty

- **Typy** – bitmapové, vektorové.
- **Uložení** – u X serveru, font server.
- **Adresa font serveru** – `tcp/aisa:7100`.
- **Nastavení** – příkaz `xset`.
- **XLFD** – řetězec, popisující font.
- **Příkazy** – `xlsfonts`, `fslsfonts`, `xfontsel`.

Obrázky

- **Pixmap, Bitmap**
- **Datové formáty** – XPM, XBM.
- **Uložení** – na X serveru.
- **Drawable** – okno, bitmapa nebo pixmapa.
- **Obsah** – uchováván X serverem.

Události

- **Význam** – informace o změně stavu.
- **Atributy** – čas, okno, souřadnice, případně další.
- **Typy** – stisk klávesy, pohyb myši, změna geometrie okna, změna mapování okna, změna viditelnosti, žádost o překreslení a další.

Window manager

- **Správce oken**
- **Dekorace oken** – potomků kořenového okna.
- **Komunikace** – seznam vlastností (properties) okna.
- **Spravovaná okna** – přímí potomci kořenového okna.

Programování

- **X protokol**
- **Xlib** – v podstatě C rozhraní k X protokolu.
- **Toolkity** – knihovny objektů. Xt, Xaw, Motif, Gtk+, Qt, XForms, Tk.
- **Uživatelská rozhraní** – aplikace a knihovny, postavené nad nějakým toolkitem (GNOME, KDE, CDE).
- **Vizuální prostředky** – vTk, Glade.

Lokalizace

- **Fonty** – existují; někdy nelze nastavit.
- **Klávesnice** – přes XKB. Ne všechny aplikace podporují.
- **Texty** – věc existence překladu.

Spuštění X

- **Z konzoly** – `startx`, `xinit`.
- **Startovací skripty** – `~/.xinitrc`.
- **Přihlášení do grafického prostředí** – Display manager (`xdm`, `gdm`).
- **XDMCP** – komunikace mezi X serverem a display managerem.
- **Startovací skripty** – `~/.xsession`.

Firewally

- **V širším smyslu** – soubor opatření k ochraně sítě proti útokům z prostoru mimo tuto síť.
- **V užším smyslu** – počítač zapojený do dvou či více sítí, který provádí filtrování dat a autentizaci uživatelů nebo strojů.

Typické nasazení

- **Vstup do Internetu** – ochrana před útokem z Internetu; kontrola přístupu lokálních uživatelů na Internet.
- **Ochrana sítě serverů** – ochrana před lokálními uživateli i před útokem z Internetu.

Typy firewallů

- **Aplikační brána** – na aplikační vrstvě.
- **Packetový filtr** – na síťové vrstvě.
- **Stavová inspekce** (stateful inspection) – kombinace obou přístupů.

Další vlastnosti

- **Network address translation (NAT)** – packetový filtr, rozdělení zátěže, atd. Některé protokoly (FTP) vyžadují aplikační bránu.
- **IP Masquerading** – speciální případ NAT – many-to-one. Kombinace packetového filtru a aplikační brány.
- **Bridging firewall** – SunScreen EFS – pracuje na linkové vrstvě.

Aplikační brána

- **Aplikační vrstva** – autentizuje se každé spojení zvlášť.
- **Proxy**
- \+ **Možnost kontroly na úrovni uživatelů**
- \+ **Není potřeba zasahovat do jádra**
- − **Nutnost úpravy aplikací**
- − **Nutnost zvláštního programu pro každou aplikaci**

Proč firewall?

- **Umožní monitorovat provoz na síti.**
- **Dovolí přesněji specifikovat práva jednotlivých uživatelů.**
- **Jeden bod přístupu do chráněné sítě.**
- **Odstíní známé i neznámé chyby v implementaci síťových služeb na vnitřních počítačích před útoky zvenku**
- **Znemožní zmapování vnitřní sítě.**

Packetový filtr

- **Síťová vrstva** – posuzuje každý packet zvlášť.
- − **Autentizace podle IP adres** – těžko lze dělat podle uživatelů.
- \+ **Transparentní** – není potřeba upravovat aplikace.
- \+ **Rychlý** – vše (většina) se odehrává uvnitř jádra.
- \+ **Bezstavový** – chod aplikací neovlivní reboot firewallu.
- \+ **Nenáročný** – lze vyrobit mini-firewall na jedné disketě.

Demilitarizovaná zóna

- **Samostatná síť**
- **(Částečně) přístupná z Internetu**
- **Částečně přístupná z vnitřní sítě**
- **Servery pro vnější služby**

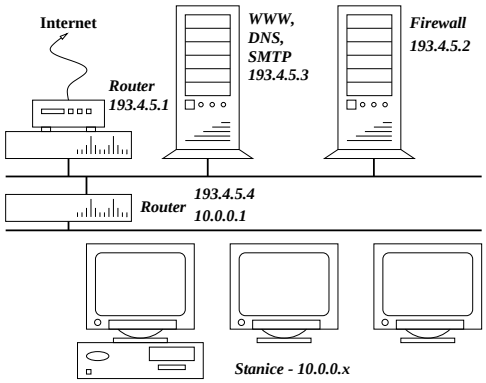
Nevýhody firewallu

- **Přístup na síť není vždy trasparentní**
- **Náklady na hardware**
- **Podcenění útoku zevnitř**

Obcházení firewallu

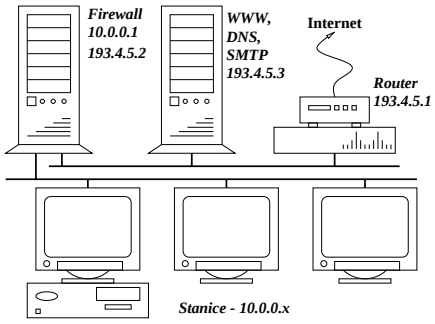
- **Firewall propouští data** $\Longrightarrow$ často je možno zevnitř vytvořit tunelový spoj ven.
- **Připojení se přímo do vnitřní sítě** – například přes dial-up modem.

Jednoportové zapojení



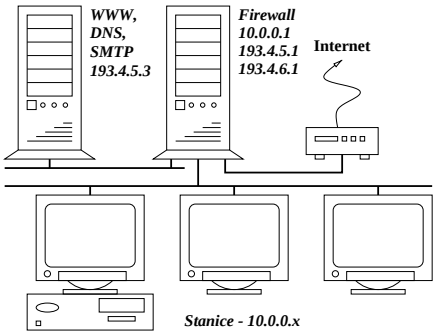
- **Použití** – obvykle v kombinaci s packetovým filtrem na routerech.
- **Aplikační brána**

Dvouportové zapojení



- **Demilitarizovaná zóna** – před firewallem. Jsou zde servery pro služby poskytované směrem do Internetu.
- **Firewall** – funguje v kombinaci s packetovým filtrem na routeru.

Tříportové zapojení



- **Demilitarizovaná zóna** – také chráněna firewallem. Firewall je obvykle typu packetový filtr.

Linux – `netfilter`

- **Netfilter** – framework pro klasifikaci, filtrování a úpravu packetů.
- **WWW** – `netfilter.samba.org`
- **Modulární architektura**
- **Pět míst, kde lze zachytit datagram**

IP Tables

- **Filtrování/úprava packetů**
- **Překlad adres** – zdrojové, cílové, případně i porty.
- **Sledování spojení** – stav packetu (`NEW`, `ESTABLISHED`, `RELATED`, `INVALID`).
- **Tabulky** – `filter`, `nat`, `mangle`.
- **Předdefinované řetězce** – podle přípojných bodů.
- **Uživatelské řetězce pravidel**
- **Pravidlo** – popis packetu, akce.
- **Packet odpovídá pravidlu** $\Longrightarrow$ provede se akce.
- **Politika řetězce** – implicitní akce.
- **Akce** – `ACCEPT`, `DROP`, `REJECT`, `RETURN`, `MASQUERADE`, `DNAT`, `SNAT`, `LOG`, předání řízení jinému řetězci a jiné.

Cesta packetu jádrem