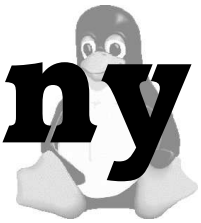
01-02/2001

Linuxové noviny



Úvodem

David Häring

A je tady první číslo Linuxových novin v novém miléniu. Doufejme, že letošní rok bude pro Linuxové noviny úspěšnější než ten loňský a také Vám čtenářům přejeme, i když trošku se zpožděním, co možná nejpříjemnější vstup do roku 2001.

V tomto čísle se dovíte mimo jiné jak nakonfigurovat síťový super-server inetd, podíváme se na monitorování zátěže, vrátíme se k problematice scanování portů a samozřejmě nechybí rubrika „Zasmáli jsme se“.

Linuxové noviny se také neobejdou bez Vašich příspěvků. Příspěvky můžete posílat jako obvykle na adresu redakce (1). ■

1 noviny@linux.cz
mailto:noviny@linux.cz

Ochrana před scanováním portů II: Scanlogd

David Häring, 10, prosince 2000

Scanlogd je dalším z nástrojů určených pro detekci scanování portů, podobně jako např. **Portsentry**, se kterým jsme se seznámili v minulém čísle Linuxových novin. Na tento článek nyní volně navazujeme a zde se již nebudeme zabývat definicemi „portscanu“ či běžnými metodami scanování portů; stručný úvod do problematiky čtenář nalezne v předchozím článku.

Scanlogd umožňuje monitorovat buď lokální systém (zpravidla přes „raw socket“), anebo i provoz celé lokální sítě (k čemuž využívá knihovny **libpcap** (1), **libnet** (2) a **libnids** (3)). Domovskou stránku projektu nalezneme na serveru Openwall (4).

Instalace a konfigurace

Instalace scanlogd je velmi jednoduchá. Pokud scanlogd instalujeme za účelem monitorování lokálního systému, kompilaci provedeme příkazem `make linux`. V tomto případě bude scanlogd používat pouze „raw socket“. Pokud chceme monitorovat provoz v lokální síti, použijeme pro sestavení příkaz `make libnids`, v tomto případě bude scanlogd používat knihovny libpcap a libnids (které musíme předtím instalovat a nastavit příslušné cesty v Makefile). Není vhodné používat pouze libpcap bez libnids (příkaz `make libpcap`), protože pak scanlogd nebude moci pracovat s fragmentovanými pakety.

Scanlogd nepoužívá konfigurační soubory, konfigurace se provádí před kompilací ruční editací hlavičkového souboru `params.h`, kde lze nastavit zejména následující položky:

- SCANLOGD_USER: uživatel, pod kterým scanlogd poběží (je záhodno vytvořit pro provozování scanlogd zvláštní účet)
- SCANLOGD_PROMISC: zde volíme, zda bude síťové zařízení pracovat v promiskuitním režimu (tedy bude přijímat veškerou komunikaci, nejen tu která je určena danému zařízení). Pokud monitorujeme pouze provoz lokálního systému, ponecháme hodnotu 0.
- PORT_WEIGHT_PRIV, PORT_WEIGHT_HIGH: zde nastavíme závažnost pokusů o scan privilegovaných portů (porty nižší než 1024 může otevřít pouze aplikace běžící s oprávněním uživatele root) a neprivilegovaných portů. Scanu privilegovaných portů se standardně přikládá větší význam.
- SCAN_MIN_COUNT: číslo říká kolik portů musí být scanováno ze stejného zdroje během určitého časového úseku, aby byl scan zaznamenán do logu.
- SCAN_DELAY_THRESHOLD: maximální časová prodleva, která může uplynout mezi scany jednotlivých portů, aby byl scan zaznamenán do logu.
- LOG_COUNT_THRESHOLD: maximální počet scanů během časového úseku, které budou zaznamenány do logu. Ochrana proti zahlcení systémového logu.
- položka SYSLOG_IDENT, položka SYSLOG_FACILITY a položka SYSLOG_LEVEL: nastavení logování (odpovídajícím způsobem je třeba nakonfigurovat také syslog).

Závažnost scanu je dána počtem scanovaných privilegovaných portů vynásobených jejich váhou (PORT_WEIGHT_PRIV) plus počet scanovaných neprivilegovaných portů vynásobených jejich váhou (PORT_WEIGHT_HIGH). Scan je považován za významný (a je zaznamenán) tehdy, je-li závažnost rovná anebo větší než hodnota SCAN_MIN_COUNT vynásobená PORT_WEIGHT_PRIV.

Použití

Při kompilaci máme možnost zvolit uživatele, pod kterým scanlogd poběží. Založíme tedy zvolený uživatelský účet (na účet není třeba se přihlašovat, můžeme tedy přihlášení zakázat). Pokud chceme, aby byl scanlogd spuštěn automaticky po startu systému, můžeme jej přidat do startovacích skriptů, např. `rc.local`).

Záznam scanu vypadá přibližně následovně:

```
Dec 10 14:24:19 server scanlogd:\
  11.22.33.44:45048 to 11.22.33.45 ports
16, 18, ..., FsrPaUxy, TOS 00, TTL 59 @14:24:19
```

V logu je uváděn čas, zdrojová a cílová IP adresa, seznam scanovaných portů, použité nastavení příznaků paketů a hodnoty TOS a TTL. V uvedeném příkladu byl proveden „Xmas“ scan (používá pakety s nastavenými příznaky

FIN, PUSH a URG; proto jsou v logu příznaky F, P a U vyznačeny velkými písmeny) na porty 19-21. Scanlogd má limitovanou velikost záznamu, který se do logu zapíše, pokud nám nestačí, tak tato položka je opět při kompilaci nastavitelná v souboru `params.h` (pamatujme ale na možnost případného zaplnění logu).

Srovnání s Portsentry

Scanlogd na rozdíl od Portsentry scany pouze detekuje a loguje, neumožňuje na scany reagovat modifikací pravidel firewallu či tcp_wrappers nebo spouštěním definovaných akcí. Dále scanlogd analyzuje pouze protokol TCP, zatímco portsentry umí detekovat i UDP scany. Na druhou stranu se portsentry nechrání proti zaplnění systémového logu a je určen pouze pro monitorování jednotlivých systémů, neumožňuje monitorovat provoz lokální sítě.

Závěrem

Scanlogd je jednoduchý nástroj pro monitorování TCP port scanů lokálního počítače nebo i okolní sítě, která je plně dostačující pokud nám jde pouze o zaznamenání scanů. ■

1 libpcap
http://www.tcpdump.org/
2 libnet
http://www.packetfactory.net/Projects/Libnet/
3 libnids
http://www.packetfactory.net/Projects/Libnids/
4 Scanlogd
http://www.openwall.com/scanlogd/

Analýza systémových logů: Logcheck

David Häring, 2. prosince 2000

Pravidelná kontrola logů by měla být nedílnou součástí údržby každého systému, i když v praxi tomu tak bohužel mnohdy nebývá. Navíc systémové logy bývají rozsáhlé a nepřehledné, takže prohlížení takových logů řádek po řádku pochopitelně není reálné. Samozřejmě, v UN*Xových systémech máme k dispozici spoustu utilit určených pro práci s textem, které pracují s regulárními výrazy, a pro zkušenějšího administrátora není problém vytvořit krátký skript pro zpracování logů spouštěný třeba přes cron. Na místo psaní vlastních skriptů ovšem můžete také sáhnout po utilitě logcheck (1).

Nejedná se o nic jiného než o krátký skript příkazového interpretu pravidelně spouštěný z cronu, který podle Vámi zadaných klíčových slov prohledá systémový log, zapamatuje si pozici, na které skončil (takže při jeho dalším spuštění z cronu neprohlíží celý logový soubor od začátku) a výsledek pošle prostřednictvím e-mailu určené osobě. Pro zapamatování pozice, na které s analýzou logu skončil logcheck používá utilitu `logtail` (která informace ukládá v souborech „jméno_log_souboru.offset“). Logcheck kontroluje také i-uzly a velikosti log souborů, takže se může vyrovnat i s rotací logů, pokud spouštění logcheck s rotací logů synchronizujeme.

Filtrování logu probíhá následovně: nejprve jsou v logu vyhledány záznamy obsahující klíčová slova z předem definovaných seznamů s cílem vychytat důležitá hlášení, která nás zajímají. Následně je log konfrontován se seznamem klíčových slov charakterizujících obvyklá normální hlášení, která se v logu vyskytují často a nejsou zajímavá; tato hlášení jsou ignorována. Na výstupu je pak zkrácená verze systémového logu obsahující významné události (dle námi zadaných klíčových slov) a dále ostatní neobvyklé události, které nepatří mezi „obvyklé“ zprávy určené k odfiltrování (opět dle námi zadaných klíčových slov — což zaručí, že bude odfiltrováno pouze to, o co opravdu nestojíme).

Instalace sestává z editace skriptu `logcheck.sh`, kde nastavíme cesty k seznamům klíčových slov, pracovnímu adresáři, cesty k log souborům, definujeme adresu na kterou chceme výstup posílat atd. (ovšem mějme zde na paměti, že systémový log obsahuje citlivé informace o systému a neměli bychom jej posílat v textové formě přes nedůvěryhodné médium). O zbytek se obstará příkaz `make linux` — přeloží utilitu `logtail` a vše potřebné instaluje na určené místo.

Na závěr musíme cron nakonfigurovat tak, aby se logcheck spouštěl pravidelně (viz manuálová stránka crontab, crond) — tady je třeba vzít v úvahu rotaci log souborů. Po několika prvních spuštění pak „vyladíme“ seznamy klíčových slov tak, aby vyhovovaly našim potřebám. ■

1 Logcheck
http://www.psionic.com/abacus/logcheck/

Konfigurujeme inetd a tcp_wrappers

David Haring, 5. prosince 2000

Tento článek je určený především začínajícím administrátorům. Vysvětlíme zde fungování „super-serveru“ inetd a podíváme se jak inetd spolupracuje s tcp_wrappers při omezení přístupu k síťovým službám.

Síťové aplikace jsou zpravidla realizovány na principu klient-server. Server i klient jsou samostatné programy. Server je pak aplikace, která běží bez přerušení a očekává spojení od klientů, jejichž požadavky vyřizuje. Klientský program je naopak zpravidla spouštěn koncovými uživateli a po vyřízení požadavků je ukončen. Příkladem může být služba telnet. Balíček telnetu obsahuje mimo manuálových stránek programy telnetd a telnet. Program telnetd je vlastní server, který se po spuštění naváže na port 23 (což je port vyhrazený pro službu telnet) a čeká na příchozí spojení. Klient telnetu, který může být spuštěn buď na stejném počítači nebo i odjinud, po spuštění kontaktuje server cílového počítače a naváže spojení, které se po ukončení sezení ukončí.

Serverové aplikace, které v Unixových systémech běží nepřetržitě na pozadí se nazývají démony. Pokud systém poskytuje větší množství síťových služeb, běží na něm současně více takových aplikací — serverů. To může vést k plýtvání systémovými prostředky a proto většina UNIXových systémů a distribucí Linuxu používá tzv. „super-server“, což bývá zpravidla inetd. Ten funguje tak, že poslouchá na portech všech služeb, které spravuje a teprve podle příchozích požadavků na spojení spouští servery obsluhující jednotlivé služby.

Některé jednoduché služby (echo, discard, chargen, time a daytime) umí inetd obsluhovat sám, bez spouštění externích serverů. Tyto služby ale nejsou zpravidla využívány a je tudíž rozumné je vypnout.

V distribuci Red Hat bývá inetd součástí balíčku **netkit-base**. Inetd může ke kontrole a omezení přístupu

```
# interní služba time
time            stream  tcp     nowait  root            internal
time            dgram   udp     nowait  root            internal
# protokol TFTP (Trivial File Transfer Protocol), bez autentikace,
# používá se k přenosu dat zpravidla při bootování přes síť
tftp            dgram   udp     wait    tftp            /usr/sbin/in.tftpd      in.tftpd/boot/diskless
# "Bootstrap Protocol" (BOOTP), slouží k přidělování IP adres a distribuci informací
# potřebných při bootování bezdiskových stanic
bootps          dgram   udp     wait    root            /usr/sbin/bootpd        bootpd
# služba finger
finger          stream  tcp     nowait  root            /usr/sbin/in.fingerd    in.fingerd
# IDENT protokol, poskytuje informace o vlastnících spojení
auth            stream  tcp     nowait  nobody.identd   /usr/sbin/in.identd     in.identd -l -e -o
# služba talk
talk            dgram   udp     wait    root            /usr/sbin/in.talkd      in.talkd
# služba FTP
ftp             stream  tcp     nowait  root            /usr/sbin/in.ftpd       in.ftpd -l -a
```

Výpis č. 1: `/etc/inetd.conf` — server poskytuje služby ftp, finger, auth, talk, tftp, bootps a time

ke službám, které spravuje, využívat balíček **tcp_wrappers** (1).

Konfigurace

Konfiguraci inetd čte ze souboru `/etc/inetd.conf`. Každý řádek obsahuje následující položky (všechny položky musí být vyplněny): „service“, „socket type“, „protocol“, „wait/nowait“, „user[.group]“, „server program“ a „server program arguments“.

Položka „service“ udává jméno služby (viz soubor `/etc/services`, případně `/etc/rpc`).

Položka „socket type“ udává typ socketu, což je nejčastěji „stream“ (pro protokoly využívající spolehlivá spojení, např. TCP) anebo „dgram“ (pro protokoly používající datagramy, např. UDP). Detaily viz např. manuálová stránka socket(2).

Položka „protocol“ udává název protokolu („tcp“, „udp“ nebo méně často „rpc/tcp“ či „rpc/udp“.

Položka „wait/nowait“ je v případě služby používající protokol TCP vždy „nowait“. V případě služby protokolu UDP záleží na tom, zda server uvolní socket a komunikuje s klientem přes nové spojení — v tomto případě může současně běžet více instancí serveru a použijeme volbu „nowait“. V opačném případě server čte datagramy tak dlouho, dokud přicházejí a po uplynutí určité doby od přijetí posledního datagramu spojení (timeoutu) se ukončí — v tomto případě použijeme volbu „wait“ (takto funguje např. talkd, démon služby talk anebo bootpd, démon služby bootps)

Položka user obsahuje jméno uživatele (případně i skupiny), pod kterým se bude daná služba spouštět.

Položka „server program“ udává cestu k serveru dané služby. V případě interních služeb se zde uvádí „internal“.

Položka „server program arguments“ udává parametry, se kterými bude server dané služby spouštěn, s výjimkou interních služeb, kdy je tato položka prázdná. Uveďme si příklad konfigurace (`/etc/inetd.conf — server poskytuje služby ftp, finger, auth, talk, tftp, bootps a time`).

Poznámka: Z historických důvodů jména démonů určených pro spouštění přes inetd zpravidla začínají na „in.“, na rozdíl od verzí démonů určených pro samostatný („stand-alone“) provoz; proto se někdy můžeme setkat v jednom balíčku se dvěma verzemi démona. Někdy také existuje jen jedna verze démona, u které patřičným parametrem můžeme zvolit, zda budeme službu provozovat přes inetd anebo samostatně.

Omezení přístupu ke službám

Jestliže jsou síťové služby, které náš systém nabízí, určeny pouze pro omezený okruh uživatelů, je žádoucí přístup k těmto službám omezit. Pro kontrolu a omezení přístupu k síťovým službám lze využít utilitu tcpd z balíčku **tcp_wrappers**. Ta funguje tak, že inetd spouští místo serveru dané služby tcpd, který nejprve dle konfigurace rozhodne, zda požadavek přijmout a teprve potom v kladném případě spustí vlastní server dané služby. Příchozí požadavky, včetně těch odmítnutých zapisuje do systémového logu. Tcpd tedy funguje jako „zástupce démonů“.

Přístup k jednotlivým službám se řídí konfiguračními soubory `/etc/hosts.deny` a `/etc/hosts.allow`. Syntaxe je jednoduchá, každý řádek je ve tvaru:

```
daemon_list : client_list [: shell_command]
```

„daemon_list“ je seznam služeb (viz `/etc/services`). Je také možné použít „ALL“, což znamená „všechny služby“, případně konstrukci ALL EXCEPT „jméno služby“, což znamená všechny služby s výjimkou té uvedené.

„client_list“ je seznam jmen počítačů či jejich IP adres. Zde je možné použít klíčové slovo „ALL“, „LOCAL“ — odpovídá jakémukoliv jménu počítače, které neobsahuje tečku, „KNOWN“ — odpovídá jakémukoliv jménu uživatele vlastnícího příchozí spojení na vzdáleném počítači (viz protokol IDENT, RFC 1413), „UNKNOWN“ — odpovídá jakémukoliv příchozímu spojení pro které nelze získat informaci o vlastníkovi spojení dle RFC 1413 (2). I zde můžeme použít klíčové slovo „EXCEPT“ (příklad: zápis .domena.cz EXCEPT badguy.domena.cz znamená všechny stroje v doméně domena.cz s výjimkou počítače „badguy“).

Položka „shell_command“ je volitelná a slouží k vykonání zadaného příkazu v příkazovém interpretu. Příkazu lze předat jako argumenty jméno či IP adresu vzdáleného počítače, jméno uživatele vlastnícího příchozí spojení na vzdáleném počítači apod.

Dejme tomu, že náš server má poskytovat v rámci domény „domena.cz“ služby ftp a finger. Ostatní služby jsou dostupné pouze lokálním uživatelům serveru. Pak může konfigurace vypadat takto:

```
ftp      stream  tcp   nowait  root          /usr/sbin/tcpd  /usr/sbin/in.ftpd -l -a
finger   stream  tcp   nowait  root          /usr/sbin/tcpd  /usr/sbin/in.fingerd
auth     stream  tcp   nowait  nobody.identd /usr/sbin/tcpd  /usr/sbin/in.identd -l -e -o
tftp     dgram   udp   wait    tftp          /usr/sbin/tcpd  /usr/sbin/in.tftpd /boot/diskless
```

Výpis č. 2: `/etc/inetd.conf` — pro kontrolu přístupu používáme tcpd

Příklad souboru hosts.deny:

```
# standardně zakázáno všechno, přístup ke službám
# musíme explicitně povolit v hosts.allow
ALL:ALL
```

Příklad souboru hosts.access:

```
# lokální uživatelé mohou používat všechny služby
ALL: LOCAL
# IDENT pouze v rámci domény
in.identd: .domena.cz
# FINGER pouze v rámci domény
in.fingerd: .domena.cz
# FTP pouze v rámci domény,
# vyžadujeme identifikaci
# vzdáleného uživatele dle RFC 931
in.ftpd: KNOWN@.domena.cz
# TFTP pouze pro bezdiskové stanice
in.tftpd: 11.22.33.44, 11.22.33.55
```

Ještě také musíme odpovídajícím způsobem upravit soubor `inetd.conf` tak, aby pro dané služby spouštěl `tcpd`. Místo cesty k serveru služby tedy uvedeme cestu k `tcpd`, viz upravená konfigurace z předchozího příkladu: soubor `/etc/inetd.conf` — pro kontrolu přístupu používáme tcpd.

Pro kontrolu nastavení přístupu ke službám přes tcp_wrappers slouží utility tcpdcheck a tcpdmatch. Například chceme-li ověřit, zda z počítače s IP adresou 11.22.33.44 je povolen přístup ke službě tftp, kterou obsluhuje přes `inetd` spouštěný démon `in.tftpd`, použijeme `tcpdmatch`:

```
$ tcpdmatch in.tftpd 11.22.33.44
client:   address  11.22.33.44
server: process  in.tftpd
matched:  /etc/hosts.allow line 12
access:   granted
```

Alternativy inetd

Často používanou alternativou k inetd se v poslední době stává **xinetd** (3), který má proti `inetd` řadu výhod. **Xinetd** nabízí například lepší kontrolu přístupu k službám (mimo jiné možnost omezení přístupu ke službám v určitých časových intervalech), ochranu proti útokům odepřením služeb (DoS), přesměrování služeb a lepší logování.

Shrnutí, bezpečnost

Inetd spolu s tcpd umožňuje upravit přístup k některým službám poskytovaných systémem. Protože standardní instalace distribucí většinou velmi benevolentně povolí přístup k řadě služeb, z nichž mnohé zpravidla vůbec nepotřebujeme, je na místě konfiguraci ihned po instalaci prohlédnout, nepotřebné služby vypnout a přístup k používaným službám omezit podle skutečných potřeb. ■

1 Tcp wrappers
ftp://ftp.porcupine.org/pub/security/index.html
2 RFC 1413
ftp://ftp.fi.muni.cz/pub/rfc/rfc1413.txt.gz
3 Xinetd
http://www.xinetd.org/

Programování s tcp_wrappers

David Haring, 10. prosince 2000

Balíček **tcp_wrappers**(1) umožňuje sledovat a filtrovat příchozí spojení řady běžně používaných síťových služeb. Kromě utility `tcpd`, která se používá pro kontrolu přístupu ke službám ve spojení s **inetd** obsahuje tcp_wrappers i knihovnu s jednoduchým rozhraním, s jejíž pomocí lze podporu tcp_wrappers zabudovat i do jiných aplikací, které nejsou spouštěny přes „super-server“ inetd.

Tcp_wrappers povolují či zamítají přístup ke službám na základě informace o jménu a IP adrese vzdáleného systému, případně i informace o vlastníkovi příchozího spojení na vzdáleném systému. Konfigurace je uložena v souborech `/etc/hosts.allow` a `/etc/hosts.deny` a je podrobně popsána v článku o konfiguraci inetd a tcp_wrappers; dále se jí zde nebudeme zabývat. Předmětem tohoto článku bude ukázka použití knihovny **libwrap**, tedy zabudování podpory tcp_wrappers do „stand-alone“ síťových aplikací.

Pro potřeby knihovny libwrap je informace o spojení uložena ve struktuře `request_info`, kterou je potřeba naplnit potřebnými údaji. Následně aplikace zavolá některou z funkcí pro kontrolu přístupu. Ve struktuře `request_info` je zejména potřeba naplnit následující položky:

- `RQ_DAEMON`: jméno procesu (serveru), který běží na hostitelském počítači
- `RQ_CLIENT_NAME`: jméno klientského počítače
- `RQ_CLIENT_ADDR`: IP adresa klientského počítače
- `RQ_SERVER_SIN`: odkaz na strukturu sockaddr_in, která obsahuje použitou síťovou adresu a port hostitelského počítače (nutné zadat pro automatické zjištění vlastníka příchozího spojení podle RFC 1413)
- `RQ_CLIENT_SIN`: odkaz na strukturu sockaddr_in, která obsahuje použitou síťovou adresu a port klientského počítače (nutné zadat pro automatické zjištění vlastníka příchozího spojení podle RFC 1413)

Knihovna libwrap poskytuje pro manipulaci s request_info následující funkce:

- `request_init()` pro inicializaci a naplnění struktury `request_info`, se kterou pracuje `hosts_access()`
- `request_set()` aktualizace obsahu již inicializované struktury `request_info`.

Pro povolení přístupu ke službě jsou k dispozici následující dvě funkce:

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <syslog.h>

/* hlavičkový soubor tcp_wrappers */
#include <tcpd.h>


/* číslo portu serveru */
#define PORT_NUM 5000
/* syslog facility */
#define LOG_FACILITY LOG_AUTHPRIV

/* tyto proměnné jsou vyžadovány tcp_wrappers */
int allow_severity=4;
int deny_severity=4;

/* deskriptory socketů apod. */
int fd,fd2;
struct sockaddr_in serv_addr, client_addr;
int client_addr_len;
struct hostent *client_info;

/* request_info pro tcp_wrappers */
struct request_info request;

char client_name[80];

/* do_exit() zapíše chybu do logu a ukončí aplikaci */
void do_exit(char *s)
{
  syslog(LOG_WARNING,"%s, exiting.",s);
  exit(1);
}

int main (int argc, char ** argv)
{
  /* argv[0] může být absolutní cesta, potřebujeme jen jméno programu */
  if (strrchr(argv[0],'/'))
    argv[0] = strrchr(argv[0],'/')+1;

  /* nastavíme způsob logování */
  openlog(argv[0],LOG_PID,LOG_FACILITY);

  /* otevíráme socket (TCP) */
  if ((fd=socket(AF_INET,SOCK_STREAM,0))<0)
    do_exit("error in socket()");
  /* navážeme se na port */
  bzero((char *)&serv_addr,sizeof(serv_addr));
  serv_addr.sin_family=AF_INET;
  serv_addr.sin_port=htons(PORT_NUM);
  serv_addr.sin_addr.s_addr=htonl(INADDR_ANY);
  if (bind(fd,&serv_addr,sizeof(serv_addr))<0)
    do_exit("error in bind()");
  if (listen(fd,5)<0)
    do_exit("error in listen()");
```

Výpis č. 3: server.c: ukázka použití knihovny libwrap

- `hosts_access()` povolí nebo zamítne přístup na základě informací v request_info, které konfrontuje s konfigurací v souborech `hosts.allow` a `hosts.deny`. Návratová hodnota `0` znamená zamítnutí přístupu.

```
  while(1) {
    client_addr_len=sizeof(client_addr);

    /* čekáme na spojení klienta */

    if ((fd2=accept(fd,&client_addr,&client_addr_len))<0)
      do_exit("error in accept()");

    /* zjistíme a uložíme jméno klientského počítače */
    if (client_info=gethostbyaddr((char *)&(client_addr.sin_addr),\
                                  sizeof(client_addr.sin_addr),AF_INET))
      snprintf(client_name, sizeof(client_name)-1,"%s",client_info->h_name);
    else
      client_name[0]='\0';

    /* naplníme request_info */
    /*   pokud zadáme i RQ_SERVER_SIN a RQ_CLIENT_SIN, tcp_wrappers  */
    /*   se pokusí identifikovat vlastníka příchozího spojení         */

    request_init(&request, RQ_DAEMON, argv[0],\
                 RQ_SERVER_SIN, &serv_addr, RQ_CLIENT_SIN, &client_addr,\
                 RQ_CLIENT_ADDR, inet_ntoa(client_addr.sin_addr),\
                 RQ_CLIENT_NAME, client_name, 0);

    /* ověříme, zda je přístup povolen */

        if (!hosts_access(&request)) {
          /* spojení zamítnuto tcp_wrappers, zapíšeme do logu */
          /* jméno vlastníka příchozího spojení je dostupné přes eval_user() */
          syslog(deny_severity, "rejected connection from %s (%s@%s)",\
                 inet_ntoa(client_addr.sin_addr),eval_user(&request),client_name);
          /* spojení ukončíme */
          close(fd2);
        }
        else {
          /* spojení povoleno tcp_wrappers, zapíšeme do logu */
          syslog(allow_severity, "accepted connection from %s (%s@%s)",
                 inet_ntoa(client_addr.sin_addr),eval_user(&request),client_name);
          /* tady bychom zpracovali požadavek klienta */
          /* spojení ukončíme */
          close(fd2);
        }
  }
}

/* překlad: gcc server.c -lwrap -lnsl */
```

Výpis č. 3: server.c: ukázka použití knihovny libwrap (pokračování)

Před voláním hosts_access() je třeba inicializovat strukturu request_info.

- `hosts_ctl()` povolí nebo zamítne přístup k službě na základě zadaných parametrů. Ve skutečnosti volá `request_init()` a následně `hosts_access()`. Na rozdíl od přímého použití `request_init()` a `hosts_access()` neumožňuje automatickou identifikaci vlastníka příchozího spojení a je na nás abychom tuto položku doplnili — pokud ji nechceme použít, doplníme `STRING_UNKNOWN`. Návratová hodnota 0 znamená zamítnutí přístupu.

Rozhraní knihovny libwrap je popsáno v manuálové stránce `hosts_access(3)`, konfigurace přístupu k službám tcp_wrappers v `hosts.access(5)`. Rovněž zdrojové kódy **tcp_wrappers** jsou velmi přehledné a komentované.

Uveďme příklad jednoduchého iterativního serveru, komunikujícího přes TCP. Příklad server.c: ukázka použití knihovny libwrap uvádí použití `hosts_access()` s využitím automatické identifikace vlastníka příchozího spojení knihovnou libwrap.

Shrnutí

S pomocí tcp_wrappers můžeme jednoduchým způsobem kontrolovat a omezit přístup k síťovým službám. Tento způsob je běžně používán nejen v souvislosti se službami

```
   procs                  memory    swap         io     system         cpu
 r  b  w   swpd   free   buff  cache  si  so    bi    bo   in    cs  us  sy  id
13  0  0      0   6500 541636 143576   0   0     3     5    1     4   0   6   3
16  0  0      0   6496 541636 143576   0   0     0     0 1856 24114  12  31  57
18  0  0      0   6512 541636 143576   0   0     1   560 1778 21169   9  32  60
20  0  0      0   6580 541636 143576   0   0     0     0 1565 21616  11  37  52
65  0  0      0   6888 541636 142732   0   0     0    10 1722 23836  10  45  46
```

Výpis č. 4: výstup příkazu vmstat -n 1 5

spouštěnými přes inetd, ale je využíván i v řadě jiných síťových služeb jako je např. ssh či portmap (NFS apod.). Knihovna libwrap umožňuje jednoduchým způsobem tuto funkčnost začlenit do libovolných síťových aplikací. ■

1 Tcp_wrappers
ftp://ftp.porcupine.org/pub/security/index.html

Monitorování zátěže

David Häring, 25. prosince 2000

Asi bychom si sotva dokázali představit provozování jakéhokoliv serveru bez pomůcek pro sledování zdraví a zátěže systému. Sledování vytížení procesoru, síťových rozhraní nebo diskových subsystémů, hledání „úzkého hrdla“ a ladění konfigurace hardware či software systému pomocí k tomu určených nástrojů patří k běžné praxi systémových administrátorů. V tomto článku se podíváme na několik běžných utilit sloužících k „online“ i „offline“ monitorování stavu systému a to zejména na utilitu vmstat, která je součástí balíčku procps a dále sar a iostat, které jsou součástí balíčku sysstat. Interaktivní nástroje pro sledování zátěže jako **top**, **gtop**, **kim**, **xosview** a další nejsou předmětem tohoto článku.

Vmstat

Vmstat periodicky vypisuje informace o počtu běžících či zablokovaných procesů, stavu paměti, stránkování, I/O operacích blokových zařízení, počtu přerušení, počtu přepnutí kontextu a vytížení procesoru. Je součástí balíčku procps. Vmstat čte informace z příslušných souborů v /proc a pro jeho spuštění nepotřebujeme žádná privilegia. Můžeme zadat 2 parametry, a to prodlevu mezi jednotlivými výpisy a počet výpisů. Jako příklad uveďme výstup příkazu vmstat -n 1 5. První řádek výpisu uvádí průměrné hodnoty sledovaných parametrů od restartu systému (s výjimkou statistiky procesů a paměti, ta je vždy aktuální).

Co znamenají jednotlivé položky?

procs:

- r: počet procesů připravených k běhu (ve výpisu utility ”top” odpovídá stavu R)
- b: počet zablokovaných procesů (ve výpisu utility **top** stav D)
- w: počet procesů připravených k běhu ale odložených (odswapovaných)

memory:

- swpd: využitý odkládací prostor (swap)
- free: volná paměť
- buff: volná paměť využitá jako vyrovnávací „buffer“ pro zápisy na disk. V případě potřeby je uvolněna.
- cache: volná paměť využitá jako vyrovnávací „cache“ pro čtení (read(), mmap() atd.). V případě potřeby je uvolněna.

swap:

- si: objem paměti načtené z odkládacího prostoru v kB/s (swapu)
- so: objem paměti odložené v kB/s (odswapované) na disk.

io:

- bi: počet bloků načtených z disků (blokových zařízení)
- bo: počet bloků zapsaných na disky (bloková zařízení)

system:

- in: počet přerušení za sekundu. Žádostí o přerušení dávají periferní zařízení najevo, že došlo k události, kterou je třeba ošetřit – např. signalizují dokončení I/O operace apod. Pokud jádro obdrží žádost o přerušení, přeruší vykonávání procesu, uloží jeho stav, ošetří přerušení, obnoví stav procesu a pokračuje v jeho vykonávání. V unixových systémech jsou tedy přerušení obsluhována v kontextu právě běžícího procesu.
- cs: počet přepnutí kontextu za sekundu. Přepnutím kontextu se rozumí přerušení vykonávání jednoho procesu, uložení stavu procesu a pokračování ve vykonávání jiného procesu.

cpu:

- us: čas procesoru (v %) strávený v režimu uživatele
- sy: čas procesoru (v %) strávený v režimu jádra. V unixových systémech mohou procesy běžet ve dvou režimech: v režimu jádra a v režimu uživatele. Proces běžící v uživatelském režimu má přístup pouze ke svým datům zatímco proces v režimu jádra má přístup i k datovým strukturám jádra. Proces vstupuje do režimu jádra např. v případě, kdy vykonává volání jádra.
- id: nevyužitý čas procesoru

Sar

Utilitu sar (z angl. „System Activity Reporter“ nalezneme v téměř každém UN*Xu a stejně tak existuje i její linuxová varianta — je obsažena v balíčku **sysstat** (1).

Sar sbírá mimo jiné data o vytížení procesorů, I/O operacích, stránkování, zpracování přerušení, vytížení síťových rozhraní, obsazení paměti, swapu, přepnutí kontextu, počtu nově vytvářených procesů. Balíček **sysstat** obsahuje utility `sar`, `sadc` a skripty `sa1`, `sa2`.

```
Linux 2.2.18-RAID (server.domena.cz)      01/09/01

16:34:57          CPU     %user     %nice   %system     %idle
16:35:02          all      9.30     35.10     30.60     25.00
16:35:07          all      9.20     36.60     29.70     24.50
16:35:12          all      9.50     37.40     28.30     24.80
16:35:17          all      8.70     37.80     27.40     26.10
16:35:22          all      8.40     38.80     25.50     27.30
Average:          all      9.02     37.14     28.30     25.54
```

Výpis č. 5: příklad výstupu příkazu sar 5 5

```
Linux 2.2.18 (server.domena.cz)         01/09/01

avg-cpu:  %user   %nice    %sys   %idle
          11.01    1.04   11.04   12.91
Disks:         tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
hdisk0        4.16         2.11         6.00    5164034   12632280
hdisk1        6.02         0.07        11.12    1869432   23470406
hdisk2        0.00         0.00         0.00          0          0
hdisk3        0.00         0.00         0.00          0          0

avg-cpu:  %user   %nice    %sys   %idle
          16.50    0.00   47.85   35.65
Disks:         tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
hdisk0        3.10         0.00         6.20          0        124
hdisk1        6.60         0.00        13.20          0        264
hdisk2        0.00         0.00         0.00          0          0
hdisk3        0.00         0.00         0.00          0          0
```

Výpis č. 6: příklad výstupu příkazu iostat 5 2

`sar` — „sa reporter“ vypisuje hodnoty zvolených parametrů systému ve zvoleném intervalu a je určen pro přímé použití z příkazové řádky.

Utilita `sadc` — „sa data collector“ sbírá údaje o systému a tato zapisuje do určeného souboru (v binárním formátu). Sadc není určen pro přímé použití, buď je spouštěn dle potřeby utilitou sar anebo může být spouštěn v rámci pravidelně spouštěných skriptů (např. pomocí **cronu**), pokud chceme offline monitorovat stav systému.

Skript příkazového interpretu `sa1` ukládá denní systémové statistiky v binárním formátu do souboru s názvem `/var/log/sadd`, kde se řetězec „dd“ nahradí dnem v měsíci. Pro prohlížení statistik pak použijeme utilitu sar. Skript volá sadc.

Skript `sa2` je určený pro zpracování denních statistik a jejich uložení v textové podobě do souboru `/var/log/sardd`. Skript volá sar.

Podobně jako `vmstat`, `sar` akceptuje jako parametr prodlevu mezi jednotlivými měřeními/výpisy statistik a počet měření/výpisů. Sar nabízí řadu přepínačů, kterými vybíráme mezi jednotlivými typy statistik, standardně vypisuje vytížení procesorů. Nebudeme zde popisovat funkce jednotlivých přepínačů, protože jich je hodně a pouze odkážeme na manuálovou stránku.

Následující (příklad výstupu příkazu sar 5 5) vypíše zátěž procesorů pětkrát v intervalu 5 sekund. Poslední řádek vždy uvádí průměrné hodnoty statistik.

Pokud chceme nepřetržitě monitorovat stav systému, stačí pomocí **cronu** pravidelně spouštět skripty `sa1` a `sa2`, které můžeme podle potřeby upravit. Následující příklad zajistí monitorování systému v intervalu 10 minut (skript `sa1`). Statistiky budou v 1 hod. následujícího dne zpracovány a uloženy v textové podobě v souboru `/var/log/sadd` (za dd dosaďme den v měsíci). Současně budou smazány statistiky starší 7 dnů (skript `sa2`).

```
Nastavení crontab:
0 * * * * /usr/lib/sa/sa1 600 6 &
0 1 * * * /usr/lib/sa/sa2 -A &
```

Ve skriptu sa2 pouze zaměníme řádek

```
DATE=‘date +%d‘"
```

za řádek

```
DATE=‘date -d yesterday +%d‘
```

Iostat

Iostat je utilitou určenou pro sledování aktivity I/O zařízení. V současných jádrech řady 2.2 je možné sledovat pouze 4 disky a nejsou k dispozici údaje o terminálových zařízeních. Pokud požadujeme statistiky v Kb/s, je zapotřebí uplatnit patch jádra ((2) — jinak jsou k dispozici pouze ve formě čtených / zapisovaných bloků za sekundu tak, jak to ukazuje příklad výstupu příkazu iostat 5 2.

Poznámka k instalaci: Pro jádra 2.4.x existuje verze sysstatu 3.3 – vývojová, pro jádra 2.2.xx je určena stabilní verze 3.2.4. Pokud používáte jádra s podporou více procesorů (SMP), je třeba buď použít jádro 2.2.16 a vyšší anebo při kompilaci balíčku sysstat zvolit opravu chyby v ovladači sériových linek, jinak může sadc vyvolat chybu kernelu při čtení souboru `/proc/tty/driver/serial`.

```
#!/bin/bash
# použití: gvmstat.sh [prodleva] [počet měření]
# zaznamenáme statistiky pomocí vmstat(1)
DELAY=${1:-5};NUM_RECORDS=${2:-50};

vmstat -n $DELAY $[$NUM_RECORDS+1] | sed -e 1,3d -e 's/^ //' -e 's/ \+/ /g' \
> /tmp/vmstatlog.$$

# zjistíme rozsahy grafů
MAX_CS=$(cut -f 13 -d ' ' < /tmp/vmstatlog.$$ | sort -n -r | head -n 1);
MIN_CS=$(cut -f 13 -d ' ' < /tmp/vmstatlog.$$ | sort -n | head -n 1);
MAX_IO_IN=$(cut -f 11 -d ' ' < /tmp/vmstatlog.$$ | sort -n -r | head -n 1);
MIN_IO_IN=$(cut -f 11 -d ' ' < /tmp/vmstatlog.$$ | sort -n | head -n 1);
MAX_IO_OUT=$(cut -f 12 -d ' ' < /tmp/vmstatlog.$$ | sort -n -r | head -n 1);
MIN_IO_OUT=$(cut -f 12 -d ' ' < /tmp/vmstatlog.$$ | sort -n | head -n 1);
MAX_IO=$[$MAX_IO_IN+$MAX_IO_OUT];MIN_IO=$[$MIN_IO_IN+$MIN_IO_OUT];

# skript pro gnuplot
cat <<EOF | gnuplot
# nejprve obecné nastœavení
set terminal gif size 640,480              # nastavíme výstup do GIFu, rozměry
set output "/tmp/vmstatlog.$$.gif"         # kam uložit výsledný GIF
set size 1,1                               # nastavíme kreslicí plochu
set origin 0,0                             # nastavíme levý dolní roh plochy
set multiplot                              # více grafů v jednom obrázku
set grid                                   # zobrazit mřížku
set xrange [0:$NUM_RECORDS]                # nastavení rozsahu osy x
set nokey                                  # vypneme legendu

# 1. graf - vytížení procesoru (součet sloupců 14 a 15 výpisu vmstat)
set size 1,0.25                            # nastavení rozměrů grafu
set origin 0,0
set yrange [0:100]                         # rozsah osy y
set label "CPU load [%]" at screen 0.5,0.2 center       # popis grafu
plot "/tmp/vmstatlog.$$" using (\$14+\$15) with lines   # vykreslení

# 2. graf - běžící procesy (sloupec 1 výpisu vmstat)
set size 1,0.25
set origin 0,0.25
set autoscale y
set label "Runnable processes" at screen 0.5,0.45 center
plot "/tmp/vmstatlog.$$" using 1 with lines

# 3. graf - přepnutí kontextu (sloupec 13 výpisu vmstat)
ycs=($MAX_CS-$MIN_CS)/5
set size 1,0.25
set origin 0,0.5
set autoscale y
set ytics ycs
set label 1 "Context switches [cws/s]" at screen 0.5,0.7 center
set nokey
plot "/tmp/vmstatlog.$$" using 13 with lines

# 4. graf - i/o statistiky (součet sloupců 11 a 12 výpisu vmstat)
ycs=($MAX_IO-$MIN_IO)/5
set size 1,0.25
set origin 0,0.75
set autoscale y
set ytics ycs
set label "I/O activity [block/s]" at screen 0.5,0.95 center
plot "/tmp/vmstatlog.$$" using (\$11+\$12) with lines
EOF
# prohlédnutí obrázku pomocí xv
xv /tmp/vmstatlog.$$.gif
```

Výpis č. 7: ukázka skriptu pro zobrazení aktivity systému pomocí vmstat

```
#! /bin/bash

# ukázka skriptu pro zobrazení aktivity systému pomocí sar
#
# použití: gsar.sh [prodleva] [počet měření]

# zaznamenáme statistiky pomocí sar(1)
DELAY=${1:-1};NUM_RECORDS=${2:-5};

sar -o /tmp/sarlog$$ $DELAY $[$NUM_RECORDS]
sar -f /tmp/sarlog$$ | sed -e 1,3d -e '$d' -e 's/ \+/ /g' -e '/^$/{N
d
}' > /tmp/sarlog$$.cpu
sar -w -f /tmp/sarlog$$ | sed -e 1,3d -e '$d' -e 's/ \+/ /g' -e '/^$/{N
d
}' > /tmp/sarlog$$.csw
sar -r -f /tmp/sarlog$$ | sed -e 1,3d -e '$d' -e 's/ \+/ /g' -e '/^$/{N
d
}' > /tmp/sarlog$$.mem
sar -b -f /tmp/sarlog$$ | sed -e 1,3d -e '$d' -e 's/ \+/ /g' -e '/^$/{N
d
}' > /tmp/sarlog$$.io

# zjistíme rozsahy grafů
MAX_CS=$(cut -f 2 -d ' ' < /tmp/sarlog$$.csw | sort -n -r | head -n 1);
MAX_IO_IN=$(cut -f 5 -d ' ' < /tmp/sarlog$$.io | sort -n -r | head -n 1);
MAX_IO_OUT=$(cut -f 6 -d ' ' < /tmp/sarlog$$.io | sort -n -r | head -n 1);
MAX_IO=`echo "$MAX_IO_IN+$MAX_IO_OUT" | bc`;
MEM_TOTAL=`grep MemTotal /proc/meminfo | sed 's/[a-zA-Z: ]//g'`;
START_TIME=$(head -n 1 /tmp/sarlog$$.csw | cut -f 1 -d ' ');
STOP_TIME=$(tail -n 1 /tmp/sarlog$$.csw | cut -f 1 -d ' ')

# skript pro gnuplot
cat <<EOF | gnuplot

# nejprve obecné nastavení
set terminal gif size 640,480;
set output "/tmp/sarlog$$.gif"
set size 1,1;
set origin 0,0;
set multiplot
set grid
set xrange [0:$NUM_RECORDS]
set nokey

# 1. graf - vytížení procesoru
set size 1,0.25; set origin 0,0
set yrange [0:100]
set xdata time; set timefmt "%H:%M:%S"
set xrange ["$START_TIME":"$STOP_TIME"]
set format x "%H:%M:%S"
set label "CPU activity [%]" at screen 0.5,0.2 center
plot "/tmp/sarlog$$.cpu" using 1:(\$3+\$4+\$5) with lines

# 2. graf - paměť (použitá celkem - (cache+ buffer))
set size 1,0.25; set origin 0,0.25
set yrange [0:$MEM_TOTAL]; set ytics 0,$MEM_TOTAL/5,$MEM_TOTAL
set xdata time; set timefmt "%H:%M:%S"
set xrange ["$START_TIME":"$STOP_TIME"]
set format x "%H:%M:%S"
set label "Used memory [kb]" at screen 0.5,0.45 center
plot "/tmp/sarlog$$.mem" using 1:(\$3-(\$6+\$7)) with lines
```

Výpis č. 8: ukázka skriptu pro zobrazení aktivity systému pomocí sar

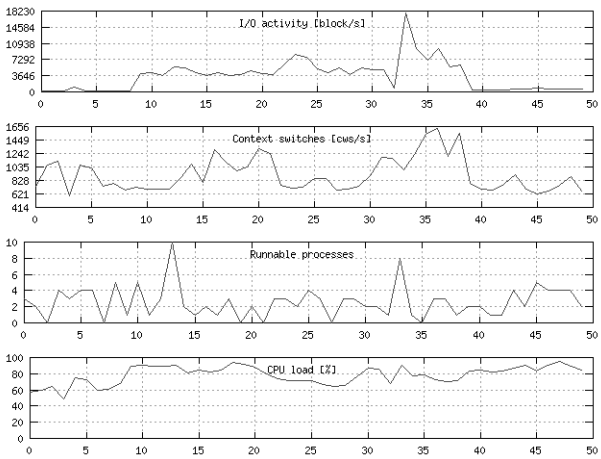
```
# 3. graf - přepnutí kontextu
set size 1,0.25; set origin 0,0.5
set autoscale ymax; set ytics $MAX_CS/5
set xdata time; set timefmt "%H:%M:%S"
set xrange ["$START_TIME":"$STOP_TIME"]
set format x "%H:%M:%S"
set label 1 "Context switches [csw/s]" at screen 0.5,0.7 center
plot "/tmp/sarlog$$.csw" using 1:2 with lines


# 4. graf - i/o statistiky
set size 1,0.25; set origin 0,0.75
set autoscale ymax; set ytics $MAX_IO/5
set xdata time; set timefmt "%H:%M:%S"
set xrange ["$START_TIME":"$STOP_TIME"]
set format x "%H:%M:%S"
set label "I/O activity [blocks/s]" at screen 0.5,0.95 center
plot "/tmp/sarlog$$.io" using 1:(\$5+\$6) with lines

EOF


# display using xv
xv /tmp/sarlog$$.gif
```

Výpis č. 8: ukázka skriptu pro zobrazení aktivity systému pomocí sar (pokračování)

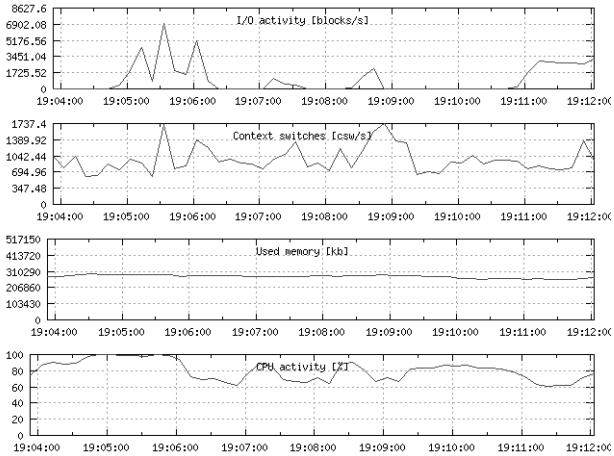


graf. výstup skriptu gvmstat.sh

Grafické znázornění

Teď už víme, jak aktivitu systému zaznamenat, zbývá tedy statistiky prezentovat nějakým přehledným grafickým způsobem. Existuje řada nástrojů, které umožňují data offline graficky zpracovat, můžeme použít například balíček **Gnuplot**(3), který je běžnou součástí distribucí.

Gnuplot je nástroj, který umožňuje tvorbu běžných typů grafů a podporuje výstup do řady grafických formátů. Je ovládaný z příkazové řádky a lze jej využít pro „offline“ zpracování dat. Používání gnuplotu však není předmětem tohoto článku, zde pouze pro inspiraci uvedeme dva příklady skriptů, které pomocí utilit vmstat (skript gvmstat.sh — ukázka skriptu pro zobrazení aktivity systému pomocí vmstat) a sar (skript gsar.sh — ukázka skriptu pro zobrazení aktivity systému pomocí sar) zaznamenají zátěž systému, pomocí gnuplotu zátěž systému vykreslí a výsledný obrázek uloží ve formátu gif. Grafické výstupy skriptu gvmstat.sh a gsar.sh ukazují přiložené obrázky.



graf. výstup skriptu gsar.sh

■

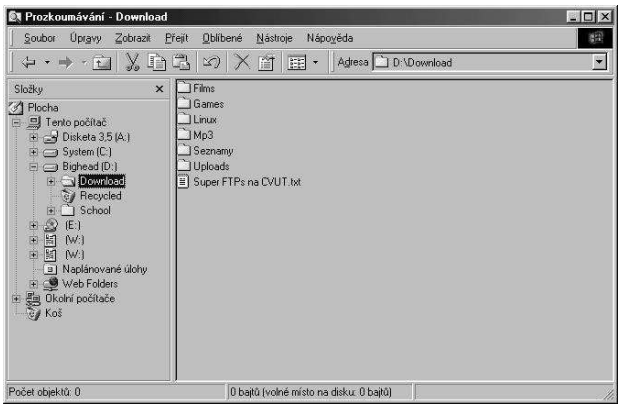
1 Sysstat
http://perso.wanadoo.fr/sebastien.godard/
2 iopatch
http://perso.wanadoo.fr/sebastien.godard/iopatch-2.2.16-v4.tar.gz
3 Gnuplot
http://www.gnuplot.org

Zasmáli jsme se!

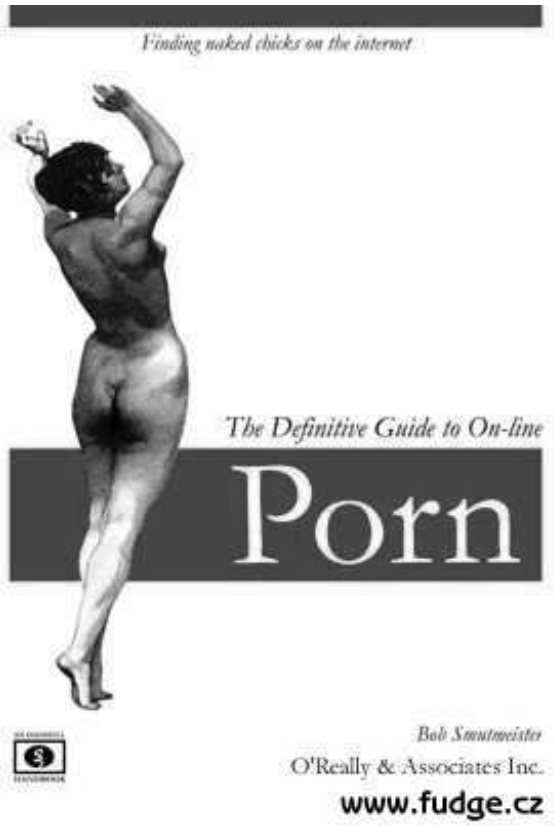
David Häring

Na závěr pro Vás máme opět několik vtípků a postřehů pro zasmání. Tentokrát mezi nimi nechybí ani tip pro Vaši knihovničku.

Některé operační systémy již od nepaměti umožňují klonovat hardware za běhu a tak uživatelům přinášejí obrovské finanční úspory. Bohužel ne vždy se to zcela povede, například v exploreru na obrázku přibyly dva nové disky — bohužel jsou téhož jména a nelze na ně zapisovat :-)



Tak tato kniha od nakladelství O'Really by Vám rozhodně v knihovničce chybět neměla …



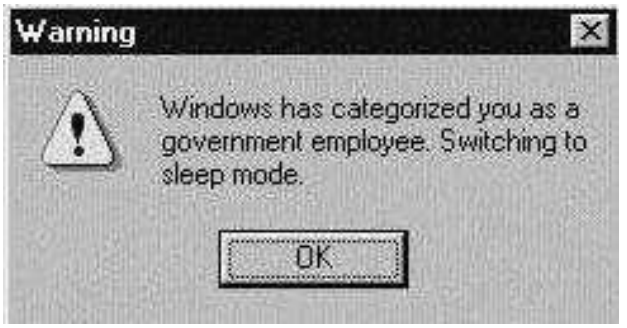
A jeden krátký recept na zbohatnutí:

> Víte proč je Bill tak bohatý? Protože za chyby se platí.

I takhle může vypadat administrace systému:

```
[laureck@chronos mpgcut]$ ./mpgcut\
   test.mpg -n 3
segmentation fault
[laureck@chronos mpgcut]$ shit!
shit!: command not found
[laureck@chronos mpgcut]$ you talkin
   to me?
you: not found
[laureck@chronos mpgcut]$ REVENGE!
REVENGE: command not found
[laureck@chronos mpgcut]$ su
Password:
[root@chronos mpgcut]# last chance son...
   behave
wtmp begins Fri Dec  1 09:56:25 2000
[root@chronos mpgcut]# rm -rf /
(no dir) system halted.
```

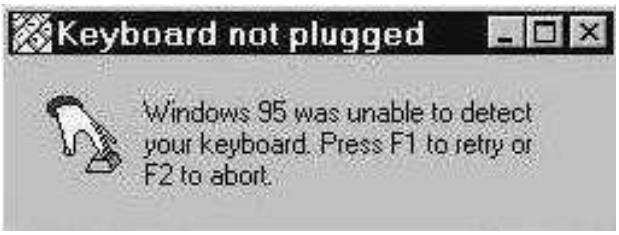
Rozhodně není pravdou, že by Windows nerozlišovaly skupiny uživatelů - jenom ta měřítka jsou trochu jiná …



Windows dbají, aby Váš harware byl „up to date“:

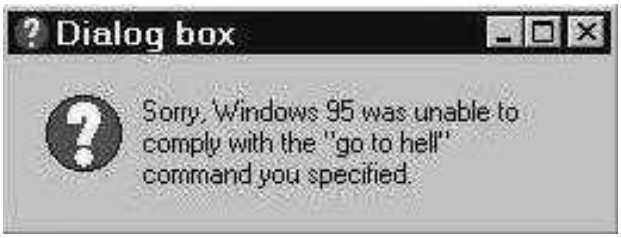
Nudíte se? Nový Random Error (tm) engine Vás probudí:

Problémy s detekcí klávesnice? MS Wizard vám poradí.



Máte problémy s tiskem? Printer Wizard je definitivně vyřeší.

Některým příkazům prostě operační systémy rozumět nechtějí:



Je po vánocích, všichni jsme přibrali a i počítače musejí hubnout …

Windows Vás chraní před předpracovaností. Déle než deset hodin v kuse byste u počítače rozhodně sedět neměli — a jestli to tak děláte, dobře Vám tak:

Linuxové noviny a jejich šíření

Linuxové noviny vydává České sdružení uživatelů operačního systému Linux (1) pro své příznivce a sympatizanty. Vlastníkem autorských práv k tomuto textu jako celku je Pavel Janík ml. (Pavel.Janik@linux.cz). Autorská práva k jednotlivým článkům zůstávají jejich autorům.

Tento text může být šířen a tištěn bez omezení. Pokud použijete část některého článku zde uveřejněného v jiných dílech, musíte uvést jméno autora a číslo, ve kterém byl článek uveřejněn.

Linuxové noviny jsou otevřeny každému, kdo by chtěl našim čtenářům sdělit něco zajímavého. Příspěvky (ve formátu čistého textu v kódování ISO 8859-2) posílejte na adresu (2). Autor nemá nárok na finanční odměnu a souhlasí s podmínkami uvedenými v tomto odstavci. Vydavatelé si vyhrazují právo rozhodnout, zda Váš příspěvek uveřejní, či nikoli.

Registrované známky použité v tomto textu jsou majetkem jejich vlastníků.

Chtěl bych poděkovat společnosti SuSE CR, s.r.o (3), že nám umožňuje i nadále pracovat na Linuxových novinách.

Linuxové noviny jsou k dispozici také ve formátu HTML na adrese (4). ■

1 České sdružení uživatelů operačního systému Linux
http://www.linux.cz/czlug
2 Adresa redakce
mailto:noviny@linux.cz
3 SuSE CR, s.r.o.
http://www.suse.cz/
4 Linuxové noviny ve formátu HTML
http://www.linux.cz/noviny

Šéfredaktor: Pavel Janík ml.
mailto:Pavel.Janik@linux.cz
zástupce šéfredaktora: David Häring
mailto:dave@ibp.cz
sazba: Ondřej Koala Vácha
mailto:koala@informatics.muni.cz
jazykové korekce: Bohumil Chalupa
mailto:bochal@met.mff.cuni.cz
překlady: Hanuš Adler
mailto:had@articon.cz
převod do HTML: Pavel Juran
mailto:juran@proca.cz