Wprowadzenie do systemów unixowych

Unix, GNU, Linux

Grzegorz J. Nalepa

AGH w Krakowie

wiosna 2015

Wykład: Historia i filozofia

Plan wykładu I

MOTTO I

Those who
do not understand Unix
are condemned
to reinvent it,
poorly.

Komputer I

Komputer *służy do*... przetwarzania danych

Składa się z 3 podstawowych elementów:

- ALU/CPU, itp. – *przetwarza* dane
- pamięć – *przechowuje* dane
- interfejsy – *przekazują* dane do/z komputera

Zadania systemu operacyjnego I

System operacyjny (OS, SO) jest podstawowym oprogramowaniem systemów komputerowych.
Niektóre z zadań współczesnych OS to:

- udostępnianie zasobów sprzętowych programom użytkowym
- abstrahowanie od specyfiki sprzętu
- dostarczenie aplikacjom różnorakich usług wysokiego poziomu, np.: bezpieczeństwo, spójność danych, komunikacja sieciowa
- wiele innych zależnych od specyfiki...

Podstawowe pojęcia I

PLIK

1. przechwowyanie danych,
2. komunikacja, przekazywanie danych z/do systemu
3. wszystko jest plikiem
4. pliki istnieją w *systemie plików*

PROCES

1. coś co przetwarza dane
2. powiązane z plikami i systemem plików

Plan punktu: Unix – rys historyczny

Unix – rys historyczny

Unix – rys historyczny I

- 1965, Multics (Multiplexed Information and Computing Service), Bell Labs, GE,
- 1969, Unix, Bell Labs, Ken Thompson, Dennis Ritchie, 1969, powstanie systemu
- 1971, pierwsza wersja,
- 1974, wersja czwarta, w C, (Brian Kernighan, Dennis Ritchie),
- 1984, UCB, 4.2BSD, TCP/IP
- 1989, System V Release 4 (SVR4),
- 1993, 4.4BSD
- 1983, GNU, Richard M. Stallman
- 1985, FSF, Richard M. Stallman
- ok. 1990, kompletne środowisko systemu GNU,
- 1991/2, Linux, Linus B. Torvalds,
- 1994 v1.0, 1995, v1.2, 1996, v2.0,
- 1999, v2.2,

Unix – rys historyczny II

- 2001, v2.4,
- 2003, v2.6
- 2011, v3.0, v3.2
- GNU/Linux
- BSDs: Free, Net, Open
- GNU/Hurd
- 2007,9, Android

Historia-odnośniki

- `http://www.bell-labs.com/history/unix`
- `http://www.unix.org/what_is_unix/history_timeline.html`
- `http://www.linuxhq.com/kernel`

Plan punktu: Unix a Linux

Unix a Linux

Linux, Unix, Hurd I

- System Linux wywodzi się z systemu Unix. Wykorzystuje środowisko systemu GNU i jądro Linux.
- Jest czasem określany mianem „klonu” Uniksa. W związku z tym można mówić o podobieństwach i różnicach pomiędzy nimi.
- GNU/Hurd jest pełnym systemem GNU – bez jądra Linux, projekt jest w fazie beta. Działająca dystrybucja oparta o Debian/GNU Hurd.

Podobieństwa Unix/Linux I

- praktycznie identyczna architektura,
- standaryzacja API (libC),
- zmierzanie do zgodności z otwartymi standardami (np. POSIX),
- przenaszalność kodu źródłowego,
- uniwersalność i skalowalność,
- Linux – „nowy” Unix?

Różnice Unix/Linux I

- projekt GNU – *GNU Is NOT Unix!*,
- licencjonowanie i dostępność GPL,
- otwartość i niezależność od korporacji,
- unowocześnienie wielu komponentów,
- praca na dowolnym sprzęcie (kilkanaście platform sprzętowych!),
- wsparcie dla najnowocześniejszych technologii (PC i Enterprise, Embedded, Mobile),
- bardzo szybki i dynamiczny rozwój.

Linux w Internecie I

- Linux Weekly News `lwn.net`
- Linux Online `www.linux.org`
- Linux.com `www.linux.com`
- Linux Kernel `www.kernel.org`
- Linux Gazette `www.linuxgazette.com`
- OS News `www.osnews.com`

Plan punktu: Standaryzacja Unixa

Standaryzacja Unixa

Standardy I

GNU/Linux zmierza do zgodności do zgodności ze specyfikacją Unixa i otwartymi standardami, w tym POSIX.
Poza tym są właściwe GNU/Linuksowi wysiłki standaryzacyjne: FHS i LSB (`http://www.linuxbase.org`).

Inne Unixy I

Komercyjne

- SUN Solaris (Open 2005) ? (Oracle)
- IBM AIX
- QNX
- RIP: HP HP-UX (RIP?), IRIX (RIP?)
- *Minix* – Protoplasta Linuxa, “dydaktyczny” Unix na x86.

BSD

- Free, Net, Open
- Apple OS X (NeXT+FreeBSD+Mach)

Inne OS I

- systemy specjalizowane, na zamówienie
- DOS (MS/Caldera/Free)
- “rodzina Windows”
- systemy wbudowane!!! np. Symbian
- systemy badawcze/hobbistyczne np. Haiku, Syllable
- ...

Plan punktu: Free Software

Free Software


FREE SOFTWARE
FOUNDATION
www.fsf.org

4 swobody

Wolne oprogramowanie (ang. Free Software)

„Wolne oprogramowanie" odnosi się do prawa użytkowników do swobodnego uruchamiania, kopiowania, rozpowszechniania, analizowania, zmian i ulepszania programów.
`http://www.gnu.org/philosophy/free-sw.html`

Cztery swobody

Dokładniej, mówimy o czterech rodzajach wolności użytkowników programu:

- wolność uruchamiania programu, w dowolnym celu (*wolność 0*),
- wolność analizowania, jak program działa, i dostosowywania go do swoich potrzeb (*wolność 1*). Warunkiem koniecznym jest tu dostęp do kodu źródłowego.
- wolność rozpowszechniania kopii, byście mogli pomóc sąsiadom (wolność 2)
- wolność udoskonalania programu i publicznego rozpowszechniania własnych ulepszeń, dzięki czemu może z nich skorzystać cała społeczność (*wolność 3*). Warunkiem koniecznym jest tu dostęp do kodu źródłowego.

Free Software I

- *free software* `www.fsf.org/philosophy/free-sw.html`
- *free as freedom NOT „zero price”/as beer*
- po polsku: *wolne*, *wolnodostępne*
- autorem koncepcji jest Richard M. Stallman,
- założyciel *Free Software Foundation* `www.fsf.org`
- twórca Projektu GNU `www.gnu.org`
- RMS jest pomysłodawcą licencji *GNU General Public License*
- inne licencje *free* to np. licencja BSD
- systemy *free* to GNU/Hurd, GNU/Linux, Free/Net/Open/BSD Unix
- prawdowpodobnie jedyna w pełni *wolna* dystrybucja Linuksa to Debian/GNU Linux

Plan punktu: Opensource Software

Opensource Software

Opensource Software I

- terminu *free software* nie należy mylić z *opensource software*
- *opensource* wywodzi się z *Debian Free Software Guidelines*
- jednym z pomysłodawców *opensource* (`www.opensource.org`) jest ESR
- *opensource* kładzie nacisk na *dostępność* kodu źródłowego,
- nie uwypukla – tak jak *free software* – prawa do jego modyfikowania i rozpowszechniania.
- we *free* vs. *open* nie chodzi o kryteria inżynierskie, ale o *etyczne*.
- używanie *non-free* software jest czymś *ZŁYM*! (RMS)

Plan punktu: Dystrybucje systemu

Dystrybucje systemu
- Dystrybucje Linuxa

Dlaczego dystrybucje Linuxa? I

W okresie gdy nie było dystrybucji (np. 1990-92):

- każdy sam musiał „składać” system,
- musiał być ekspertem,
- wymagany był dostęp do sieci,
- występowały niezgodności programów,
- brakowało możliwości uaktualniania systemu.

Pierwsze dystrybucje I

Dawały one między innymi:

- zbiór działających ze sobą komponentów systemu,
- były dostępne na jednym nośniku (dyskietki, CDROM),
- zawierały wskazówki i uwagi twórców dystrybucji,
- pozwalały na uruchomienie systemu nie tylko ekspertom.

Funkcje współczesnej dystrybucji I

- wspomaganie procesu instalowania systemu w różnych konfiguracjach,
- zawieranie mechanizmów jego uaktualniania,
- pozwalanie na automatyczne uzgadnianie wersji instalowanych programów,
- dawanie stabilnej platformy pracy,
- integrowanie oprogramowania,
- zapewnianie administratorowi wsparcia technicznego w podstawowym zakresie.

Elementy dystrybucji I

- jądro systemu Linux (ang. *kernel*),
- GNU: LibC, narzędzia GNU, powłoki, i inne,
- kompilator GNU GCC,
- oprogramowanie systemowe przeniesione z Unixa, takie jak Init, Syslog, Cron, itd.,
- system X11: XOrg, XFree86,
- progr. do zarządzania pakietami,
- program instalacyjny,
- dodatkowe oprogramowanie.

Typy dystrybucji I

- uniwersalne wieloplatformowe i jednoplatformowe,
- specjalizowane,
- dla systemów wbudowanych.

Dystrybucje uniwersalne I

Najważniejsze dystrybucje uniwersalne to:

- Debian/GNU, Ubuntu, Knoppix
- Mandriva
- Novell: Suse Pro, Suse ES, NLD
- PLD :)
- RedHat: RHEL, Fedora, CentOS, inne
- Slackware, Slax

Dystrybucje specjalizowane I

Podstawowe kategorie to:

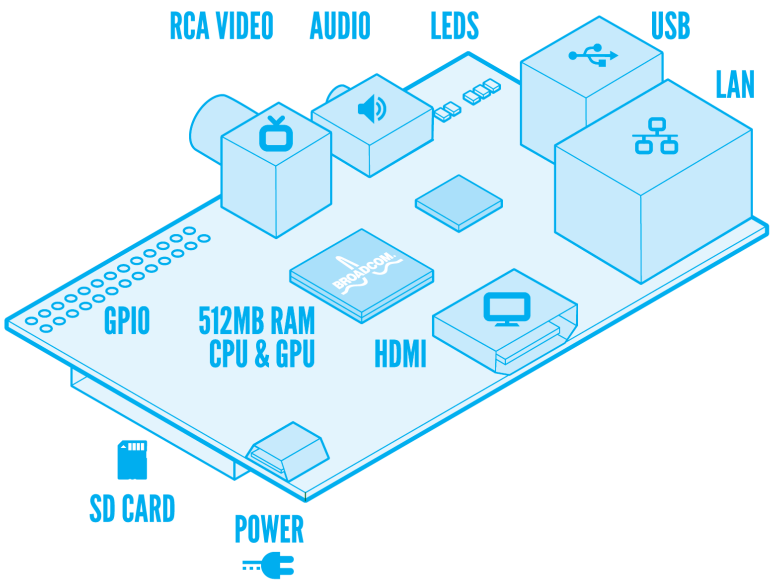
- o zwiększonym bezpieczeństwie: Nexus, Trustix, Immunix, Astaro
- miniaturowe: Trinux, Tomsrb,
- działające z CDROM/USB: Knoppix, CDLinux, Slax
- inne: LTSP.

Wiele różnych wykazów dystrybucji można znaleźć na stronach `http://www.linux.org` i `http://lwn.net`.

Linux dla systemów wbudowanych I

- Wykaz dystrybucji, można znaleźć na stronie `http://lwn.net/Distributions/#embed`.
- Przykład: *uClinux*: dystrybucja dla CPU bez MMU, dostępna na dzień dzisiejszy na szereg architektur wbudowanych, np: m68k/Freescale/DragonBall/Coldfire, Arm, itp.
- Android (2009)
- Moblin, Maemo, MeeGo...
- ARM CPU jako platforma

RASPBERRY PI MODEL B



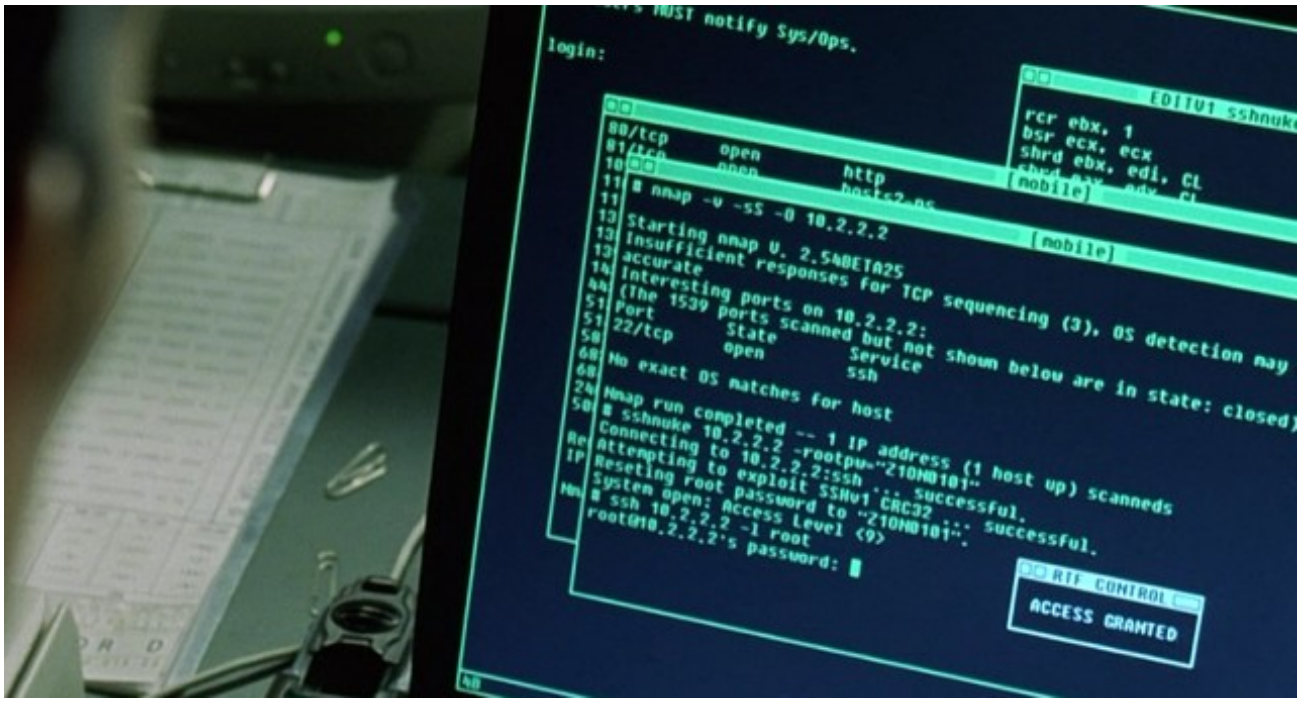
Dystrybucje systemów BSD

- Współczesne BSD: Free, Net, Open
- Na dzień dzisiejszy poza głównymi w.w. dystrybucjami są pochodne, takie jak PC-BSD, czy FreeSBIE.
- Dystrybucja BSD dostarcza analogicznych elementów jak dystrubucja GNU/Linux.
- Aplikacje dodatkowe są jedynie „portowane”
- Całość można w całości zbudować z postaci źródłowej
- Podstawową różnicą jest metoda jej rozwijania. . . konkretny system BSD *jest* dystrybucją; system operacyjny stanowi zintegrowaną całość.
- Apple OS X ;)

Dawno temu

- Slackware by Patrick Volkerding
- 1994: This is Slackware Linux 2.1.0.: This version contains libc 4.5.26, Linux kernel 1.1.59, (plus source for many other versions in the source tree, including version 0.01), and XFree86 3.1.
- 1995:Welcome to the April 1995 Slackware 2.2 CDROM from Walnut Creek CDROM! This is Slackware Linux 2.2.0. This version contains libc 4.6.27, Linux kernel 1.2.1, and XFree86 3.1.1.
- 1996: This is Slackware Linux 3.1.0 (Slackware96). This version contains the 2.0.0 Linux kernel, plus recent versions of these (and other) software packages. . .

Matrix


login:
80/tcp open http
nmap -v -sS -O 10.2.2.2
Starting nmap V. 2.54BETA25
Insufficient responses for TCP sequencing (3), OS detection may
accurate
Interesting ports on 10.2.2.2:
(The 1539 ports scanned but not shown below are in state: closed)
Port State Service
22/tcp open ssh
No exact OS matches for host
Nmap run completed -- 1 IP address (1 host up) scanned
sshnuke 10.2.2.2 -rootpw="Z1ONM101"
Connecting to 10.2.2.2:ssh ... successful.
Attempting to exploit SSHv1 CRC32 ... successful.
Reseting root password to "Z1ONM101".
System open: Access Level (9)
ssh 10.2.2.2 -l root
root@10.2.2.2's password:
ACCESS GRANTED

Tron


$ whoami
flynn
$ uname -a
SolarOS 4.0.1 Generic_50203-02 sun4m i386
Unknown.Unknown
$ login -n root
Login incorrect
login: backdoor
No home directory specified in password file!
Logging in with home=/
bin/history
488 cd /opt/LLL/controller/laser/
489 vi LLLSDLaserControl.c
490 make
491 make install
492 ./sanity_check
493 ./configure -o test.cfg
494 vi test.cfg
495 vi ~/last_will_and_testament.txt
496 cat /proc/meminfo
497 ps -a -x -u
498 kill -9 2207
499 kill 2208
500 ps -a -x -u
501 touch /opt/LLL/run/ok
502 LLLSDLaserControl -ok 1
#

Plan punktu: FAQ

FAQ

FAQ I

- Q: Jakiego GNU/Linuxa/BSD używać?
  A: Tego który działa ;)
- Q: Jakiego Pan używa?
  A: Debian/GNU, Ubuntu, Free/OpenBSD
- Q: Jakiego będziemy używać na laboratorium?
  A: Debian/GNU, Ubuntu?
- Q: Mam komputer z innym system operacyjnym i mam problem...
  A: W rzeczy samej!
- Q: Czy mogę zainstalować GNU/Linux razem z innym systemem operacyjnym?
  A: Ale po co? (VirtualBox)

debian

MOTTO I

Those who
do not understand Unix
are condemned
to reinvent it,
poorly.

Wykład: Architektura systemów unixowych

Plan wykładu I

Architektura systemu

Struktura modelu bezpieczeństwa

Konto administratora

Unixowy system plików

Opis systemu plików w Linuxie

Dokumentacja Linuxa

Motto I

A droga wiedzie w przód i w przód,
Skąd się zaczęła, tuż za progiem -
I w dal przede mną mknie na wschód,
A ja wciąż za nią - tak, jak mogę...
Skorymi stopy za nią w ślad –
Aż w szerszą sie rozpłynie drogę,
Gdzie strumień licznych dróg już wpadł...
A potem dokąd? – rzec nie mogę.

Plan punktu: Architektura systemu

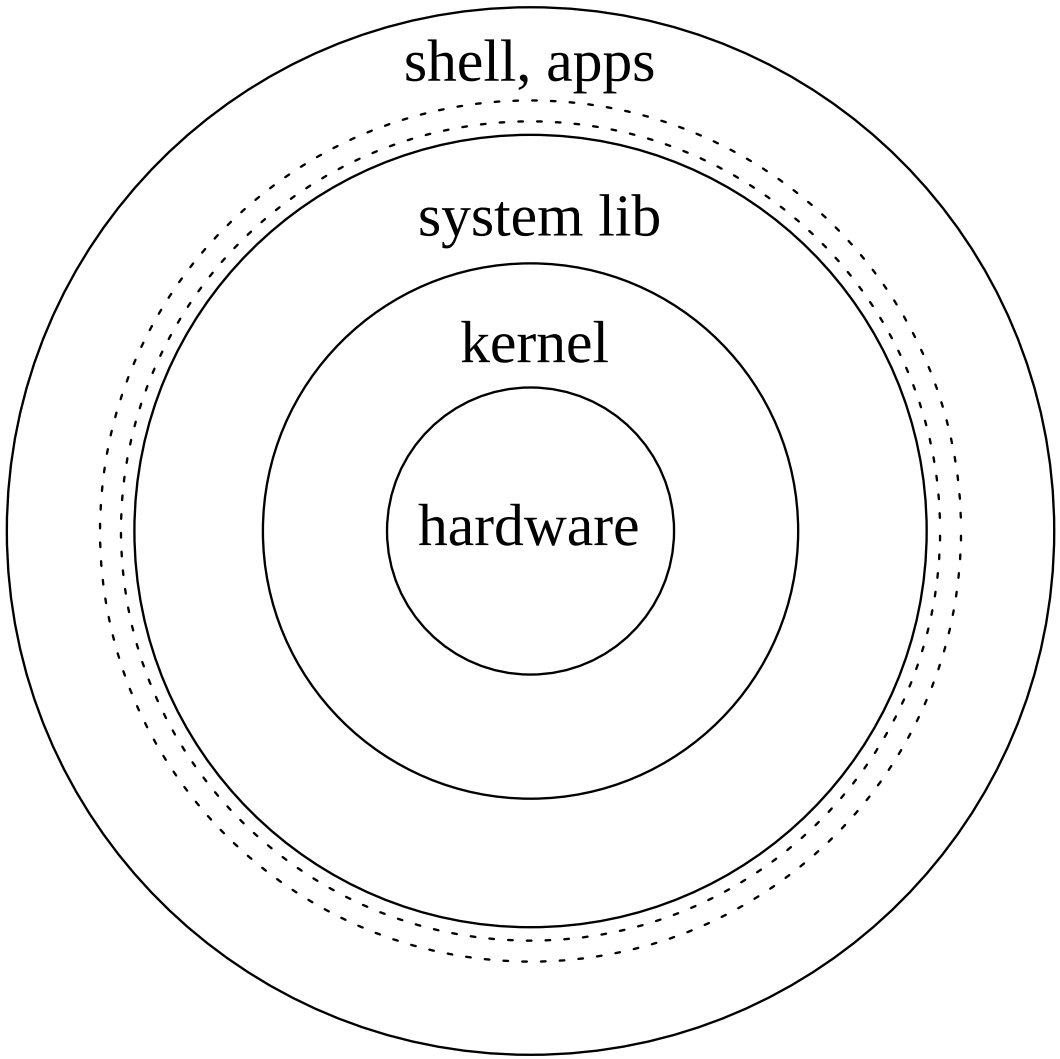
Architektura systemu

Elementy systemu typu GNU/Linux I

Na współczesny system GNU/Linux składają się:

- jądro Linux (Linus Torvalds),
- biblioteka systemowa GNU Lib C (Richard M. Stallman, FSF),
- oprogramowanie systemowe wywodzące się głównie z systemów BSD Unix,
- powłoki systemowe i podstawowe narzędzia GNU (FSF),
- system XOrg, XFree86, który jest implementacją systemu X Window.

Architektura I



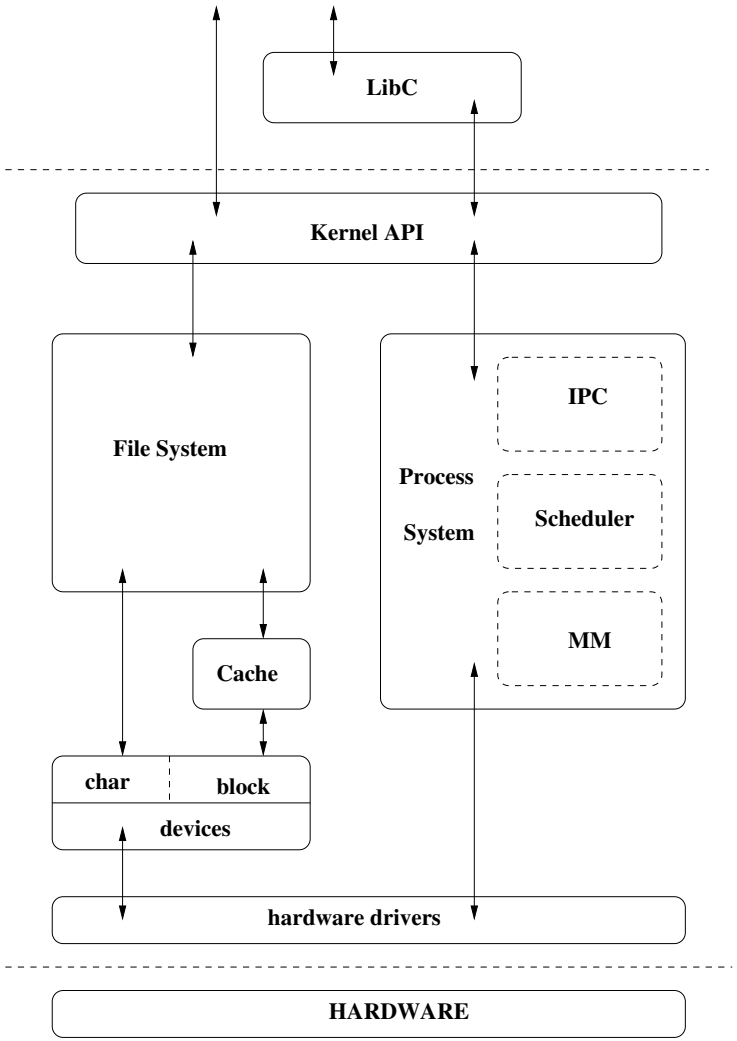
Jądro systemu I

Jądro (ang. *kernel*) jest kluczowym elementem systemu. Jego główne role to:

- zarządzanie dostępem do zasobów sprzętowych w tym procesora i pamięci,
- umożliwianie komunikacji przez urządzenia we/wy,
- kontrola pracy współbieżnie pracujących programów,
- kontrola pracy równocześnie pracujących użytkowników.

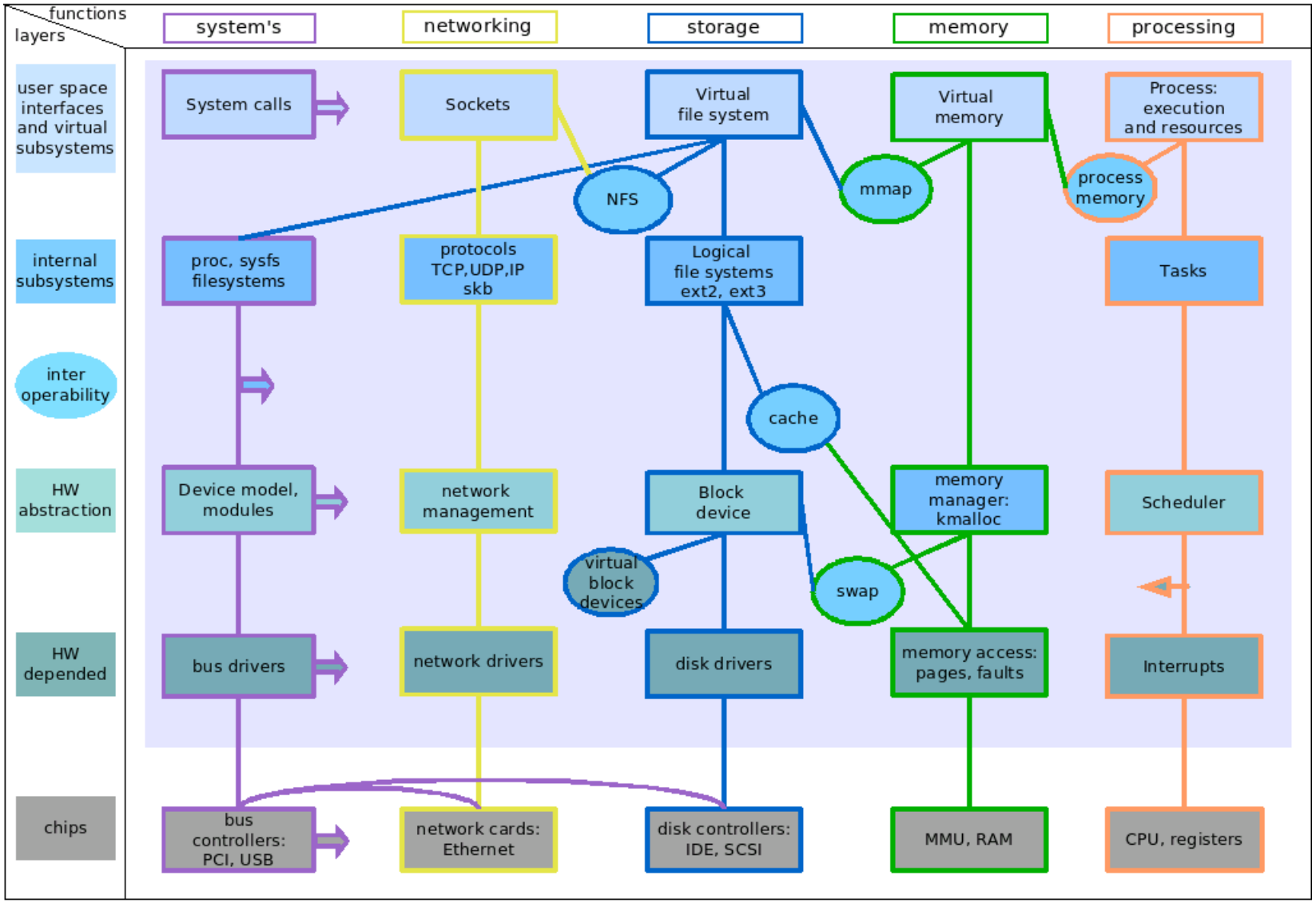
Podstawowe elementy jądra to:

- podsystem plików (w tym kilkupoziomowe buforowanie danych)
- podsystem zarządzania procesami (w tym pamięcią, przydziałem CPU)
- podsystem sterowania sprzętem
- interfejs funkcji systemowych



M.J. Bach, Budowa systemu operacyjnego UNIX

Simplified Linux kernel diagram in form of a matrix map



Designed with OpenOffice.org by (cc) (by-nc-sa) Constantine Shulyupin, www.linuxdriver.co.il

©*Constantine Shulyupin, Wikipedia.org (Creative Commons)*

Biblioteka systemowa I

Biblioteka standardowa (systemowa, LibC) jest, obok jądra, najważniejszym elementem systemu.

- zapewnia interfejs programistyczny (API) do systemu operacyjnego,
- jest *zestandaryzowana* co umożliwia wysoki stopień przenaszalności kodu źródłowego,
- pośredniczy w komunikacji programów z jądrem,
- jest z punktu widzenia programów niezależna od jądra systemu.

Powłoki i aplikacje I

Z wymienionych elementów korzystają programy.

- programy komunikują się z systemem przez bibliotekę standardową,
- powłoki (ang. *shell*) są podstawowym interfejsem użytkownika do systemu,
- pozwalają użytkownikowi na uruchamianie innych programów,
- dodatkowym komponentem może być X Window, zintegrowane z TCP/IP, graficzne środowisko do tworzenia okienkowych interfejsów użytkownika.

X Window I

Dodatkowym komponentem systemu jest środowisko X Window:

- jest graficznym środowiskiem do tworzenia okienkowych interfejsów użytkownika,
- pozwala na pracę w rozproszonym, heterogenicznym środowisku,
- jest niezależne od platformy i standardowy dla większości wersji systemu Unix,
- jest ściśle zintegrowane z siecią TCP/IP,
- zostało zrealizowane w architekturze klient/serwer.

Plan punktu: Struktura modelu bezpieczeństwa

Struktura modelu bezpieczeństwa

Struktura modelu bezpieczeństwa I

Model bezpieczeństwa w Unixie Model zawiera:

- konta użytkowników,
- zróżnicowane systemy uwierzytelniania,
- ochronę zasobów na poziomie systemy plików,
- ochronę zasobów na poziomie procesów i pamięci,
- współcześnie: zastosowanie zaawansowanych technik kryptograficznych.

Plan punktu: Konto administratora

Konto administratora

Zadania administratora I

Niektóre z najważniejszych:

- administrowanie sprzętem
- administrowanie oprogramowaniem
- administrowanie kontami
- utrzymywanie bezpieczeństwa
- włączanie i wyłączanie systemu
- tworzenie i odtwarzanie kopii zapasowych
- monitorowanie pracy systemu
- kontakt z użytkownikami systemu

Przywileje konta *root* I

Z dostępem do głównego konta systemu – *root* – wiążą się pewne przywileje. Daje ono na przykład możliwość:

- czytania każdego pliku
- zapisu do każdego pliku
- uruchomiania i przerwania każdego programu

Używanie konta *root* I

Pewne podstawowe zasady związane z bezpiecznym użyciem konta, to:

- ochrona hasła
- dawanie innym użytkownikom minimalnych, wymaganych uprawnień
- praca na tym koncie powinna być związana wyłącznie z zadaniami administracyjnymi
- zmiana id. użytkownika – polecenie **su**

Podział zadań I

Podział zadań polega nie tylko na tym, że administrowaniem zajmuje się kilka osób, lecz również na stworzeniu odrębnych kont administracyjnych do różnych zadań. Zalety:

- zmniejszenie ilości pracy wykonywanej przez jedną osobę,
- dokładniejsze wykonywanie zadań,
- przejrzysty podział uprawnień,
- zmniejszone prawd. nadużycia przywilejów.

Wadą może być potencjalne zmniejszenie bezpieczeństwa systemu.

Plan punktu: Unixowy system plików

Unixowy system plików
- Pliki i katalogi
- Węzły indeksowe
- Struktura systemu plików
- Prawa dostępu
- Dowiązania
- Bity SUID i sTicky
- Pliki urządzeń

Pliki i katalogi I

- Dostęp do wszystkich zasobów systemu Unix jest realizowany przez pliki.
- Plik jest wygodną *abstrakcją*.
- Wszystko jest plikiem.
- Pliki są reprezentowane w systemie plików przez *węzły indeksowe* (ang. *i-node*).
- Katalog jest szczególnym przypadkiem pliku, który zawiera listę w postaci par: nazwa pliku, i-node.

Węzły indeksowe I

- I-node jest podstawowym elementem składowym systemu plików.
- Zawiera wszystkie informacje o pliku, poza jego nazwą (ta jest w katalogu!).
- *Wskazuje* na to, gdzie się *fizycznie* znajdują dane zawarte w pliku.
- Przechowuje daty: modyfikacji węzła *ctime*, modyfikacji (danych) pliku *mtime*, dostępu do pliku *atime*.
- Opisuje: typ i rozmiar obiektu, wraz z liczbą dowiązań (nazw).
- Określa właściciela i grupę pliku, oraz *prawa dostępu*.

Struktura systemu plików I

System plików (ang. *filesystem*) jest *logiczną* strukturą organizacji plików.

- W systemie Unix ma strukturę drzewiastą.
- Jest *zawsze tylko jedna* taka struktura.
- W związku z tym *zawsze* jest bezwzględny początek „/" systemu plików.
- Inne systemy mogą być włączane jako kolejne gałęzie.
- Katalog bieżący: . katalog nadrzędny . .
- Wszystkie pliki i katalogi mają nazwy będące ciągami znaków alfanumerycznych.
- Nazwy mogą być długie i są *case sensitive*.
- Katalogi rozdziela się znakiem / (ang. *slash*).
- *Ścieżka dostępu* pozwala na umiejscowienie pliku w strukturze systemu.
- *Pełna* (bezwzględna) ścieżka określa jego położenie względem początku drzewa, zaczyna się od /, np. */etc/passwd*.
- *Względna* ścieżka określa położenie względem katalogu bieżącego.

Polecenie **ls** I

```
ls -al
drwxr-xr-x  3 gjn  gjn   4096 Feb 26 18:54 .
drwxrwxrwt 23 root root 28672 Feb 26 18:53 ..
-rw-r--r--  1 gjn  gjn      4 Feb 26 18:53 ala
drwxr-xr-x  2 gjn  cdrom 4096 Feb 26 18:54 burego
lrwxrwxrwx  1 gjn  gjn      3 Feb 26 18:53 kota -> ala
-rw-r--r--  1 gjn  users    3 Feb 26 18:53 ma
```

Atrybuty pliku I

```
-rwxrwxrwx N uzytkownik grupa rozmiar data nazwa
```

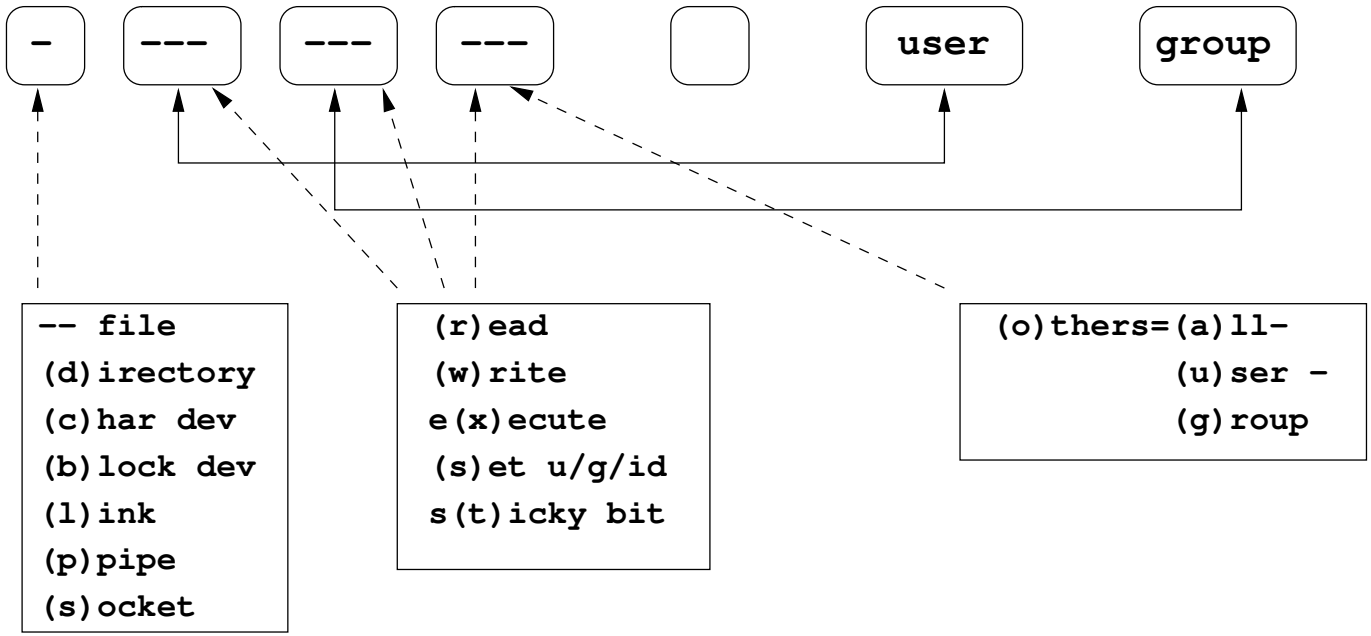
Prawa dostępu:

```
 -    rwx    rwx    rwx
TYPE USER GROUP OTHERS
```

Typy plików

```
TYPE: - d b c l p s
```

Prawa dostępu I



Rysunek: Unixowe prawa dostępu

Zmiana praw dostępu I

```
chmod [ugao][+-=][rwxst] plik
```

na przykład:

```
chmod u+x plik
chmod g-rw plik
chmod o+x,u-r,g=w plik
```

Numeryczne prawa dostępu I

```
r=4, w=2, x=1
su=4, sg=2, t=1
chmod NNNN plik
```

na przykład:

```
chmod 755 plik
chmod u=rwx,g=rx,o=rx plik
chmod 644 plik
chmod u=rw,g=r,o=r plik
chmod 44 plik
chmod 0044 plik
```

Domyślne prawa, umask I

```
$ umask
0022
$ umask -S
u=rwx,g=rx,o=rx
$ touch nowy1 ; ls -l nowy1
-rw-r--r--  1 gjn gjn 0 Feb 26 20:39 nowy1
$ umask 700 ; touch nowy2 ; ls -l nowy2
----rw-rw-  1 gjn gjn 0 Feb 26 20:40 nowy2
$ umask 077 ; touch nowy3 ; ls -l nowy3
-rw-------  1 gjn gjn 0 Feb 26 20:40 nowy3
$ umask 000 ; touch nowy4 ; ls -l nowy4
-rw-rw-rw-  1 gjn gjn 0 Feb 26 20:40 nowy4
```

Prawa do katalogów I

- *write* zapis do katalogu
- *read* odczyt zawartości (spisu plików!), **ls**
- *execute* dostęp do zawartości (plików), **cd**

Prawa do katalogów - przykład I

```
$ chmod a=rx katalog
$ ls katalog
plik
$ cd katalog
$ cd ..
$ chmod a=r katalog ; ls katalog
plik
$ cd katalog
sh: cd: katalog: Permission denied
$ chmod a=x katalog ; cd katalog
$ ls
ls: .: Permission denied
```

Zmiana właściciela I

```
chown user  plik
chgrp grupa plik
```

Zmienić właściciela może tylko root.

Dowiązania I

- Dowiązania (polecenie **ln**) pozwalają na dostęp na dostęp do pliku pod inną nazwą, ścieżką dostępu.
- Symboliczne, są oddzielnymi plikami (i-node'ami), które tylko zawierają odniesienie do pliku źródłowego.
- Sztywne (ang. *hard*), są kolejnymi fizycznymi dowiązaniami do tego samego i-node.
- Sztywne mogą być tylko w obrębie jednego systemu plików.

Bity SUID i sTicky I

Bity SUID

- W przypadku plików wykonywalnych programów zmieniają uprawnienia procesu.
- W przypadku katalogu SGID powoduje, że nowo tworzone pliki mają taką grupę jak katalog, a nie jak bieżąca grupa użytkownika.
- sTicky powoduje, że pliki w katalogu może usuwać tylko właściciel.

Pliki urządzeń I

Pliki urządzeń

- Reprezentują fizyczne (np. dysk twardy) lub logiczne (pseudoterminal) urządzenia.
- Zakłada się je poleceniem **mknod**.
- Są identyfikowane po parze liczb *major,minor*.
- Uprawnienia do nich są *krytyczne* dla działania systemu! (np. */dev/kmem*, */dev/hda*).

Pliki i Procesy I

Bazowe pojęcia

PLIK i *PROCES* to 2 najważniejsze pojęcia w Unixie.

Uprawnienia

Są one powiązane z modelem bezpieczeństwa. Uprawnienia do jednych wpływają na uprawnienia do drugich.

Plan punktu: Opis systemu plików w Linuxie

Opis systemu plików w Linuxie

FHS I

- Drzewo plików systemu plików jest zestandaryzowane.
- *Filesystem Hierarchy Standard*, autor: Daniel Quinlan (i inni) `www.pathname.com/fhs`,
- Twórcy najpopularniejszych dystrybucji Linuksa przestrzegają go.
- Standard jest obecnie częścią LSB (*Linux Standard Base* `www.linuxbase.org`).
- Docelowo ma się odnosić do innych unixowych systemów otwartych, w tym ew. do BSD

Katalogi */bin* */sbin* I

W tych dwóch katalogach znajdują się najważniejsze polecenia systemu.

- w katalogu */bin* są programy dostępne dla wszystkich użytkowników systemu, między innymi: **cat, chgrp, chmod, chown, cp, dd, kill, ln, mkdir, more, mount, mv, ps, sh, sync, tar, gzip, netstat, ping**.
- katalog */sbin* przechowuje polecenia dla administratora, na przykład: **init, mkswap, swapon, swapoff, shutdown, fdisk, fsck, mkfs, lilo, ifconfig, route**

Katalog */boot* I

W tym katalogu powinny się znajdować dwa rodzaje plików, związane ze startem systemu:

- pliki dla programu Lilo (lub innego loadera): *boot.b, chain.b, mbr.b, os2_d.b, map*
- pliki jądra: *vmlinux, vmlinuz, zImage, bzImage, System.map*

Katalog */dev* I

W tym katalogu znajdują się pliki specjalne do obsługi urządzeń, np:

- *hd** – urządzenia IDE,
- *fd** – napędy dyskietek,
- *lp** – porty równoległe,
- *ttyS** – porty szeregowe,
- *tty** – konsole (terminale),
- *pty** – pseudoterminale,
- *dsp*, midi*, mixer** – urządzenia dźwiękowe,
- *null, zero* – specjalne funkcje.

Katalog */etc* I

W tym katalogu znajdują się wszystkie pliki konfiguracyjne systemu:

- */etc/X11* – konfiguracja X Window,
- */etc/skel* – szablony plików konfiguracyjnych kont użytkowników,
- */etc/init.d*, */etc/rc.*d* – pliki konfiguracyjne Init,
- */etc/fstab* – spis systemów plików,
- */etc/inittab* – główna konfiguracja Init,
- */etc/passwd*, *group*, *shadow*, *gshadow* – informacje o kontach użytkowników,
- */etc/host**, *inetd.conf*, *resolv.conf* – plik związane z siecią TCP/IP.

Katalogi */home /root /lib* I

Te katalogi pełnią następujące funkcje:

- */home* – zawiera katalogi domowe użytkowników,
- */root* – katalog domowy administratora,
- */lib* – podstawowe biblioteki systemu.

Katalogi */mnt* */tmp* I

Poniższe katalogi pełnią funkcje specjalne:

- */mnt* – zawiera punkty montowań systemów plików, na przykład:

  ```
  /mnt/cdrom
  /mnt/floppy
  /mnt/nfs
  /mnt/samba
  ```

- */tmp* – tu przechowywane są pliki tymczasowe użytkowników i tworzone przez inne programy.

Katalog */proc* I

Ten katalog zawiera wirtualny system plików, służący do komunikacji z jądrem, np:

- */proc/cmdline* – linia poleceń jądra,
- */proc/cpuinfo* – informacje o procesorze,
- */proc/devices* – dostępne urządzenia,
- */proc/dma* – używane kanały DMA,
- */proc/filesystems* – obsługiwane systemy plików,
- */proc/interrupts* – używane przerwania,
- */proc/ioports* – używane adresy I/O,
- */proc/kcore* – obraz pamięci RAM,
- */proc/kmsg* – komunikaty jądra,
- */proc/loadavg* – obciążenie systemu,
- */proc/meminfo* – informacje o pamięci,
- */proc/modules* – załadowane moduły,
- */proc/mounts* – podmontowane systemy plików,

Katalog */proc* II

- */proc/net/* – katalog zawierający informacje o sieci,
- */proc/pci* – informacje o magistrali PCI,
- */proc/uptime* – czas od startu systemu,
- */proc/version* – pełna wersja jądra.

Od wersji jądra 2.6 przechodzi się na wykorzystywanie katalogu */sys*

Katalog */usr* I

W tej części systemu plików znajduje się przeważnie większość oprogramowania:

- */usr/X11R6* – katalog zawiera pliki dystrybucji systemu X Window,
- */usr/bin* – większość programów wykonywalnych, dostępnych dla wszystkich użytkowników,
- */usr/include* – pliki nagłówkowe dla standardowych bibliotek,
- */usr/lib* – biblioteki dzielone,

Katalog */usr* I

- */usr/local* – osobne drzewo plików, o strukturze podobnej do */usr*; tu administrator umieszcza programy spoza dystrybucji,
- */usr/man* – pliki podręcznika man,
- */usr/sbin* – polecenia dostępne dla administratora systemu,
- */usr/share* – dodatkowe dane (moduły) dla programów, niezależne od architektury sprzętowej (np. dane graficzne),
- */usr/src* – kody źródłowe programów,

Katalog */var* I

Zawiera dane które często się zmieniają:

- */var/catman* – sformatowane strony man,
- */var/lib* – tymczasowe pliki danych, np. czcionki dla systemu TEX,
- */var/local* – odpowiednik */var*, dla */usr/local*,
- */var/lock* – pliki spejcalne *lock files*,
- */var/log* – pliki logów systemowych,
- */var/run* – pliki dotyczące stanu programów,
- */var/spool* – kolejki plików różnego rodzaju,
- */var/tmp* – katalog dla plików tymczasowych wszystkich użytkowników, przechowywanych dłużej niż w */tmp*.

Plan punktu: Dokumentacja Linuxa

Dokumentacja Linuxa

Podręczniki Man Pages I

Jest to zbiór tradycyjnych podręczników dostępnych w każdym systemie Unix. Większość poleceń systemowych ma swoją stronę podręcznika. W podręcznikach omówione są również funkcje systemowe, formaty plików i inne. Podręczniki są przeważnie dość zwięzłe. Dzielą się na numerowane kategorie (ang. *sections*). Dostęp przez polecenie **man**.

Kategorie stron *man* I

- 1 Executable programs or shell commands
- 2 System calls
- 3 Library calls
- 4 Special files
- 5 File formats and conventions
- 6 Games
- 7 Macro packages and conventions
- 8 System administration commands
- 9 Kernel routines

man – użycie I

Najczęściej używane wywołania polecenia **man**

- **man hasło** – pierwsza strona związana z `hasłem`,
- **man -f hasło** – opis wszystkich stron dot. `hasła`
- **man -k słowo** – spis wszystkich stron, których nazwa lub opis zawierają podane `słowa`,
- **man -a hasło** – wszystkie strony dot. `hasła`,
- **man n hasło** – pokazuje stronę w sekcji `n`.

Dodatkowa dokumentacja I

Katalog */usr/share/doc* zawiera dodatkową dokumentację do programów. Znajduje się ona w katalogu, którego nazwa odpowiada nazwie pakietu deb lub rpm z którego został zainstalowany program.

Dokumentacja TEXinfo I

Dokumentacja w tym formacie dotyczy najczęściej projektu GNU:

- jest bardziej obszerna i wyczerpująca niż dokumenty man,
- ma charakter podręczników i przewodników,
- format TEXinfo umożliwia wygenerowanie dokumentów w hipertekstowym formacie *info*,
- łatwo można z niej uzyskać profesjonalnie złożony dokument w postaci nadającej się do wydruku.

Opisy dystrybucji I

Dystrybucje systemu GNU/Linux są opisane w osobnych dokumentach (w */usr/share/doc*:

- w dystrybucji Debian/GNU są to: *Debian-FAQ*, *Debian-Manifesto*, *Debian-Policy*, *Debian-Social Contract*
- natomiast w przypadku Redhat: *RedHat Manual*

Dokumenty HOWTO I

Cechy tych dokumentów to między innymi:

- są tworzone przez ochotników z *The Linux Documentation Project* (`www.tldp.org`),
- występują w wielu wersjach językowych,
- każdy dokument jest dostępny w różnych formatach: sgml, dvi, ps, ascii, html,
- szczegółowo i wyczerpująco omawiają pojedyncze, określone w tytule zagadnienie.

Książki LDP I

Projekt LDP stworzył m.in. poniższe książki:

- Installation and getting started.
- Kernel Hacker's Guide.
- The Linux Kernel.
- Network administrator's guide.
- Programmer's guide.
- System administrator's guide.
- User's guide.

Wykład: Podstawy zarządzania systemem

Plan wykładu I

Uwierzytelnianie i konta użytkowników

Systemy plików

Tryby pracy systemu

Nowe rozwiązania w Linux v2.6

Motto I

DO!
or
DO NOT!
There is no TRY!

Tematyka wykładu I

- zarządzanie kontami użytkowników
- zarządzanie systemami plików
- tryby pracy i uruchamianie systemu

Administrowanie systemem I

Podstawowe zadania administratora to:

- monitorowanie systemu
- zarządanie kontami użytkowników
- instalowanie i uaktualnianie oprogramowania
- uruchamianie i zamykanie systemu
- zarządzanie systemami plików

Plan punktu: Uwierzytelnianie i konta użytkowników

Uwierzytelnianie i konta użytkowników

Zarządzanie kontami użytkowników I

- dane o użytkownikach,
- zakładanie nowego konta użytkownika,
- modyfikacja konta,
- usuwanie konta użytkownika.

Dane o użytkownikach I

- System identyfikuje użytkowników na podstawie numerów nazywanych *User ID*, *uid*.
- Dodatkowa baza danych zawiera informacje umożliwiające zamianę tych numerów na nazwy tekstowe.
- Zawiera ponadto informacje opisujące konto użytkownika i definiujące jego środowisko pracy.

Plik */etc/passwd* I

Wspomniana baza jest zawarta w pliku `/etc/passwd` i zawiera:

1. nazwa użytkownika (*Username*)
2. zaszyfrowane hasło (*Encrypted Password*)
3. identyfikator użytkownika - (*uid*)
4. identyfikator głównej grupy użytkownika
5. imię i nazwisko, lub opis konta - (*GECOS*)
6. katalog domowy (*Home Directory*)
7. powłoka użytkownika (*Login Shell*)

Grupy I

- Użytkownicy w systemie mogą być przypisywani do grup.
- Plik */etc/passwd* przechowuje informacje tylko o podstawowej grupie użytkownika.
- Informacja o przynależności użytkownika do innych grup jest przechowywana w pliku */etc/group*.
- Każdy użytkownik *musi* być przynajmniej w jednej grupie, podstawowej (ang. *primary*).

Plik */etc/group* I

Każda linia tego pliku ma postać :

1. nazwa grupy
2. hasło - rzadko używane, najczęściej w tym polu znajduje się znak „*” (nie w przypadku *shadow passwords*)
3. identyfikator grupy - *gid*
4. lista użytkowników - nazwy użytkowników oddzielone przecinkami

Shadow passwords I

Plik */etc/passwd* może być czytany przez wszystkich.Może stanowić zagrożenie dla poufności haseł. Z tego powodu hasła są przeważnie przechowywane w osobnym pliku.

Shadow Passwords

System korzysta on z dodatkowego pliku */etc/shadow*, który zawiera hasła, oraz inne informacje, na przykład o okresie ważności konta. Ten plik może być czytany tylko przez administratora systemu.

Podsumowanie I

Większość istotnych informacji o użytkownikach jest przechowywana w poniższych plikach:

- */etc/passwd* – podstawowe informacje o kontach użytkowników,
- */etc/group* – podstawowe informacje o grupach użytkowników,
- */etc/shadow* – rozszerzone informacje o kontach użytkowników (np. daty ważności) i zaszyfrowane hasło (w systemie *shadow*),
- */etc/gshadow* – rozszerzone informacje o grupach użytkowników (w systemie *shadow*).

Zakładanie konta I

Podstawowe metody zakładania kont użytkowników:

- z wykorzystaniem **adduser**,
- z wykorzystaniem **useradd**,
- przy pomocy edytora.

Nowe konta – **adduser** I

- program **adduser** jest rozbudowanym narzędziem do zakładania kont,
- jest standardowo dostępny w większości dystrybucji GNU/Linuxa,
- w innych dystrybucjach prostszy **useradd**,
- umożliwia również zakładanie grup,
- można konfigurować jego pracę w pliku */etc/adduser.conf*,

Przykład **adduser** I

```
# adduser jacek
Adding user jacek...
Adding new group jacek (1004).
Adding new user jacek (1004) with group jacek.
Creating home directory /home/jacek.
Copying files from /etc/skel
Changing password for jacek
Enter the new password (minimum of 5, maximum of 8 characters)
Password changed.
Changing the user information for jacek
Enter the new value, or press return for the default
Is the information correct? [y/n] y
```

Nowe konta – **useradd** I

- program **useradd** jest prostym do zakładania kont,
- jest standardowo dostępny we wszystkich dystrybucjach GNU/Linux,
- towarzyszy mu narzędzie do zakładania grup użytkowników,
- można zmieniać jego pracę z linii poleceń.
- przy pomocy opcji -**D** można dokonać konfiguracji domyślnych wartości parametrów programu (katalog, powłoka, itp.).

Ponieważ polecenie nie ustawia hasła, należy skorzystać z **passwd**. Podobnie ma się rzecz z innymi parametrami konta. Jeżeli program nie został uprzednio skonfigurowany przy pomocy opcji -**D**, to może na przykład zajść konieczność stworzenia katalogu domowego.

Zmiana hasła I

- hasło zmienia polecenie **passwd**,
- program pyta o stare hasło i 2 razy o nowe,
- każdy użytkownik może zmieniać swoje hasło (jeżeli zna stare hasło),
- administrator może zmieniać hasło każdego użytkownika (bez znajomości starego hasła).

Nowa grupa I

Przeważnie wystarczy stworzyć grupę przy pomocy:

```
# addgroup grupa
```

Oczywiście można podać parametry dodatkowe (np. *gid*).

Dopisanie do grupy I

Poniższe wywołanie jest możliwe tylko w przypadku polecenia **adduser** (**addgroup**):

```
# addgroup jacek games
Adding user jacek to group games...
Done.
```

Ręczne zakładanie kont I

Przykład ręcznego tworzenia konta:

- dopisać do pliku */etc/passwd* linijkę:

```
jacek:*:550:550:Jacek Kowalski
        :/home/stud/jacek:/bin/tcsh
```

- dopisać do pliku */etc/group*:

```
jacek:*:550
students:*:600:jacek
```

- stworzyć katalog domowy:

```
mkdir /home/stud/jacek
```

- skopiować katalog */etc/skel*:

```
cp -R /etc/skel/{*,.*} /home/stud/jacek
```

- prawa dostępu do katalogu domowego:

Ręczne zakładanie kont II

```
chown -R jacek.students /home/stud/jacek
chmod -R u=rwX,g=rwX,o= /home/stud/jacek
```

- zmienić hasło użytkownika:

```
passwd jacek
```

Informacje o koncie I

Można je uzyskać przy pomocy poleceń takich jak **id**, **groups**, np:

```
$ id
uid=1000(gjn) gid=1000(gjn)
groups=1000(gjn),25(floppy),29(audio),
30(dip),60(games),100(users)
$ groups
gjn floppy audio dip games users
```

Modyfikacje konta I

Raz założone konto można modyfikować:

- polecenie **chfn** zmienia informacje GECOS (imię, nazwisko, itp.) o użytkowniku,
- polecenie **chsh** zmienia powłokę,
- narzędzie **usermod** modyfikuje dowolne parametry konta,
- narzędzie **groupmod** j.w. dla grupy,
- instrukcja **passwd** zakłada nowe hasło, a w systemie *shadow passwords* zmienia daty ważności konta,
- **chage** zmienia daty ważności konta.

Usuwanie kont i grup I

- konto użytkownika można usunąć przy pomocy polecenia **userdel**,
- powyższe polecenie z opcją **-r** usuwa katalog domowy użytkownika,
- polecenie **groupdel** usuwa grupy użytkowników,
- przy pomocy polecenia instrukcji **find** można odnaleźć i usunąć pliki których właścicielem jest podany użytkownik lub grupa.

Blokowanie konta I

Kilka metod blokowania dostępu do konta:

- poleceniem **passwd** z opcją **-l**,
- przez ręczną modyfikację hasła w pliku *passwd/shadow*,
- poprzez zmianę powłoki użytkownika na program nie dopuszczający do logowania.

Trzeba pamiętać, iż powyższe metody nie są skuteczne w przypadku sieciowych metod autoryzacji nie korzystających z haseł, np. *rhosts*.

Konta systemowe I

W systemie istnieje specjalna grupa kont:

- mają niskie UID, przeważnie poniżej 100,
- nie mają katalogów domowych,
- często nie należą do żadnej grupy,
- mają zablokowane logowanie – brak hasła,
- nie mają przypisanej powłoki,
- są to „wirtualni” użytkownicy służący do automatyzacji zadań administracyjnych,
- przykłady takich kont: daemon, bin, sys, man.

Inne systemy autoryzacji I

- Wraz ze zmianą potrzeb coraz częściej stosowane są modyfikacje tradycyjnego modelu.
- Modularna, warstwowo-komponentowa architektura systemu, w tym: GNU LibC NSS i system PAM, pozwalają na używanie „usług katalogowych”, np.: LDAP, NDS, WinPDC/AD, RDBMS.

Plan punktu: Systemy plików

Systemy plików

System plików I

Podstawowe informacje:

- jest to logiczna struktura organizacji danych,
- zawiera informacje o położeniu i atrybutach plików,
- przechowuje dane znajdujące się w plikach,
- może odwzorowywać lokalizację plików na napędzie fizycznym (np. twardy dysk), lub wirtualnym (np. ramdysk),
- w przypadku dysków twardych systemy plików znajdują się na partycjach,
- metadane opisujące plik przechowuje *i-node*.

System ext2 I

- podstawowy system plików w Linuksie,
- implementacja uniksowego systemu plików,
- jest to nowoczesny system plików, zapewniający między innymi automatyczną defragmentację,
- jest implementacją standardowego uniksowego systemu plików,
- towarzyszy mu pakiet wyspecjalizowanych narzędzi – e2fsprogs,
- jego rozwinięciem jest Ext3.

Zarządzanie systemami plików I

Zarządzanie systemami plików to między innym:

- zakładanie partycji na dyskach fizycznych,
- tworzenie systemów plików,
- montowanie (podłączanie),
- odmontowywanie (odłączanie),
- sprawdzanie,
- utrzymywanie wolnego miejsca.

Tworzenie partycji I

Partycje tworzy się przy pomocy programu typu **fdisk**.

1. **fdisk** jest podstawowym programem pracującym w trybie tekstowym,
2. jego rozwinięciem jest **cfdisk**,
3. inne programy tego typu są uruchamiane w czasie instalowania systemu,
4. funkcje wszystkich są podobne: zakładanie i usuwanie partycji różnego typu.

fdisk – przykład I

```
# fdisk /dev/hdb
Command (m for help): p
Disk /dev/hdb: 32 heads, 63 sectors, 827 cylinders
Units = cylinders of 2016 * 512 bytes
   Device Boot   Begin    Start       End   Blocks   Id  System
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-827): 1
Last cylinder or +size or +sizeM or +sizeK ([1]-827): +100M
Command (m for help): p
Disk /dev/hdb: 32 heads, 63 sectors, 827 cylinders
Units = cylinders of 2016 * 512 bytes
   Device Boot   Begin    Start       End   Blocks   Id  System
/dev/hdb1             1        1       102   102784+  83  Linux native
```

Nowe systemy plików I

Na istniejącej partycji można założyć nowy system plików przy pomocy polecenia **mkfs**. Użycie: `mkfs -t typ [-c] urzadzenie`

-t typ – typ systemu plików, np. *ext2*,

-c – sprawdzenie, czy na urządzeniu nie ma uszkodzonych bloków (ang. *bad blocks*) – parametr opcjonalny,

urzadzenie – urządzenie (np. partycja dysku) na której zakładany jest system plików.

mkfs – przykład I

```
# mkfs -t ext2 /dev/hdb1
mke2fs 1.10, 24-Apr-97 for EXT2 FS 0.5b, 95/08/09
Linux ext2 filesystem format
Filesystem label=
8320 inodes, 33232 blocks
1661 blocks (5.00%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
5 block groups
8192 blocks per group, 8192 fragments per group
1664 inodes per group
Superblock backups stored on blocks:
        8193, 16385, 24577, 32769
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

Montowanie I

- aby skorzystać z systemu plików, trzeba go zamontować, (podmontować)
- nowy system jest dostępny w podanym katalogu,
- dostęp do systemu plików może być regulowany przy pomocy opcji **mount**,
- w ogólnym przypadku, tylko administrator może montować systemy plików,
- polecenie wywołane bez opcji wyświetla podmontowane systemy plików.

Polecenie **mount** I

Składnia **mount**: **mount -t typ urządzenie punkt_montowania -o opcje**

typ – typ systemu plików,

urzadzenie – urządzenie na którym znajduje się system plików,

punkt_montowania – katalog w którym udostępniany jest nowy system plików,

-o opcje – różne opcje dodatkowe, np. **ro,rw,remount**; zależą również od typu systemu plików.

Przykłady – mount I

```
# mount
/dev/hda1 on / type ext2 (rw,errors=remount-ro)
proc on /proc type proc (rw)
# mount -v -t ext2 /dev/hdb2 /mnt/old -o ro
/dev/hdb2 on /mnt/old type ext2 (ro)
# mount
/dev/hda1 on / type ext2 (rw,errors=remount-ro)
proc on /proc type proc (rw)
/dev/hdb2 on /mnt/old type ext2 (ro)
```

Odmontowywanie I

- do odmontowywania służy polecenie **umount**,
- odmontowanie jest możliwe tylko wtedy, gdy żaden proces nie korzysta z plików na odmontowywanym systemie plików,
- przy odmontowaniu wystarczy podać punkt montowania lub urządzenie.

Plik */etc/fstab* I

- w tym pliku umieszcza się listę najczęściej montowanych systemów plików,
- przy montowaniu systemów plików tu wpisanych wystarczy podać samo urządzenie lub punkt montowania,
- dla poprawnego funkcjonowania systemu konieczne jest przynajmniej umieszczenie wpisu dotyczącego głównego systemu plików,
- plik składa się z linii, z których każda ma składnię podobną do opcji polecenia **mount**:
  `<file system> <mount point> <type> <options> <dump> <pass>`

Sprawdzanie I

- do sprawdzania systemów plików służy polecenie **fsck**,
- **fsck** wywołuje specjalne wersje narzędzi dla konkretnego systemu plików (np. **e2fsck**),
- **e2fsck** można również wywoływać bezpośrednio,
- systemy plików są systematycznie sprawdzane w trakcie startu systemu,
- w systemie ext2/3 jest licznik montowań, którego zmniejszenie do 0 wymusza sprawdzanie,
- użycie systemu z *journalingiem* może zmniejszyć prawdopodobieństwo utraty danych w przypadku awarii, oraz skrócić czas sprawdzania.

Opcje fsck I

`fsck -t typ [opcje] urządzenie`

typ – typ systemu plików,

urzadzenie – urządzenie na którym jest system plików,

opcje – dodatkowe opcje, np. -**A** – sprawdzanie systemów wg. */etc/fstab*; -**r** – tryb interaktywny; -**f** – wymuszenie sprawdzania dla ext2.

Użycie fsck I

```
# fsck -t ext2 -v -f /dev/hdb2
Parallelizing fsck version 1.10 (24-Apr-97)
e2fsck 1.10, 24-Apr-97 for EXT2 FS 0.5b, 95/08/09
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information

    9347 inodes used (44%)
     159 non-contiguous inodes (1.7%)
         # of inodes with ind/dind/tind blocks: 873/26/0
   74401 blocks used (88%)
       0 bad blocks

    7716 regular files
     656 directories
     356 character device files
     233 block device files
```

Użycie fsck II

```
      2 fifos
    790 links
    372 symbolic links (372 fast symbolic links)
      3 sockets
--------
  10128 files
```

Partycje swap I

- są używane jako pamięć wirtualna systemu,
- umożliwiają maksymalne zwiększenie ilości dostępnej pamięci,
- zakłada się je poleceniem: **mkswap**,
- włącza (montuje) przez: **swapon**,
- wyłącza (odmontowuje) przez: **swapoff**,
- możliwe jest również korzystanie z plików swap.

Podział systemu plików I

Zalety:

- łatwiejsze zarządzanie poszczególnymi fragmentami systemu plików,
- szybsza praca systemu – partycje mogą się znajdować na osobnych szybkich dyskach,
- zwiększone bezpieczeństwo w przypadku uszkodzeń systemu plików,
- możliwość montowania w trybie tylko do odczytu,
- ułatwione tworzenie kopii zapasowych,
- proste dołączanie nowych napędów fizycznych do systemu.

Propozycje podziału I

- */* – główne katalogi tworzą szkielet, zawierając pliki i polecenia niezbędne do startu systemu i podmontowania innych systemów plików; powinien zawierać przynajmniej: */bin /boot /dev /etc /lib /mnt /proc /root /sbin*,
- */home* – katalogi domowe użytkowników, w przypadku systemów z wieloma użytkownikami mogą mieć bardzo dużą objętość; oprócz tego są często dzielone pomiędzy maszynami,
- */usr* – ten katalog zawiera większość używanych programów, może być dzielony i używany w trybie tylko do odczytu,
- */usr/local* – ten katalog pełni podobne funkcje jak powyższy,
- */var* – oddzielenie tego, często zapisywanego, fragmentu zmniejsza obciążenie głównego systemu plików,
- */var/spool/mail* – warto oddzielić w przypadku serwera poczty elektronicznej,
- */var/spool/news* – warto oddzielić w przypadku serwera grup dyskusyjnych,
- */tmp* – jeden z najczęściej zapisywanych katalogów, jego oddzielenie zmniejsza obciążenie systemu,

Nowa generacja FS I

Około roku 2002 zostały udostępnione stabilne wersje zaawansowanych systemów plików:

- Ext3 (Tweedie,/Ts'o/Miller/Torvalds)
- ReiserFS (H. Reiser)
- XFS (SGI)

Wszystkie oferują *księgowanie* (ang. *journaling*), różnorakie optymalizacje, ew. zaawansowane funkcje dodatkowe, np. POSIX ACL.

Inne systemy plików I

- Linux obsługuje ponad 30 różnych FS.
- mogą się znajdować na dyskach fizycznych, sieciowych lub być wirtualne,
- obsługa systemów plików przez jądro jest wykonywana przez VFS (*Virtual File System*),
- różne systemy plików montuje się zmieniając opcje **typ** polecenia **mount**,
- z większością systemów plików związane są dodatkowe opcje,
- obsługa danego FS musi być wkompilowana w jądro.

Montowanie vfat I

Przykład montowania systemu plików windows 9x: **# mount -t vfat -o noexec,conv=auto,quiet /dev/hdb1 /mnt/vfat1**

-t vfat – typ systemu plików,

noexec – pliki nie będą wykonywalne,

conv=auto – automatyczne konwersja znaków końca linii w plikach tekstowych,

quiet – nie są wyświetlane ostrzeżenia dotyczące praw dostępu.

Plan punktu: Tryby pracy systemu

Tryby pracy systemu

Tryby pracy systemu I

- uruchamianie systemu,
- zatrzymywanie systemu,
- program LILO/GRUB,
- program Init.

Uruchamianie – jądro I

Uruchamianie systemu można podzielić na pewne ogólne etapy:

- start programu ładującego jądro (bootstrapu), np. **lilo**,**grub**,
- ładowanie i dekompresja jądra,
- inicjalizacja podstawowych urządzeń, których obsługa jest wkompilowana w jądro,
- sprawdzenie i podmontowanie głównego systemu plików,

Uruchamianie – Init I

Po załadowaniu jądra uruchamiany jest **init**:

- uruchomienie procesu **init** (*zawsze* PID=1),
- przeczytanie pliku konf. */etc/inittab*,
- sprawdzenie i podmontowanie pozostałych systemów plików,
- przejście na odpowiedni poziom pracy (*runlevel*),
- uruchomienie usług na funkcjonujących na tym poziomie,
- zezwolenie na logowanie się użytkowników.

Poziomy pracy systemu I

- poziom 0 oznacza zatrzymanie (*halt*),
- poziom 1 jest trybem dla jednego użytkownika (*single*) – administratora,
- poziomy 2 – 5 są różnymi trybami pracy dla wielu użytkowników,
- 6 – oznacza zatrzymanie i restart (*reboot*),
- poziomy pracy można zmieniać: **# init N**
- przeważnie *inittab* jest w *stylu* SYSV, każdemu poziomowi odpowiada katalog */etc/rcN.d* i w nim skrypty *Kxx Sxx*

Konfiguracja Init I

Konfiguracja w pliku */etc.inittab*.
Plik składa się z linii, z których każda ma następującą składnię:
`id:runlevels:action:process`

- **id** identyfikator, czteroliterowa nazwa linii,
- **runlevels** poziomy pracy na których zostanie wykonana akcja (*action*),
- **action** np.: respawn, wait, boot, initdefault, sysint, ctrlaltdel
- **process** proces który ma zostać uruchomiony, przeważnie odpowiedni skrypt startowy.

Przykład */etc/inittab* I

```
id:2:initdefault:
si::sysinit:/etc/rc.d/bcheckrc
l0:0:wait:/etc/rc.d/rc.halt
l1:1:wait:/etc/rc.d/rc.single
l2:2345:wait:/etc/rc.d/rc.multi
l6:6:wait:/etc/rc.d/rc.reboot
ca::ctrlaltdel:/sbin/shutdown -t5 -rf now
g1:23:respawn:/sbin/getty 38400 tty1
g2:23:respawn:/sbin/getty 38400 tty2
```

Uniwersalny skrypt rc I

W nowych dystrybucjach Linuksa stosuje się uniwersalny skrypt *rc*.

- pracuje na każdym poziomie od 0 – 6,
- skrypt zatrzymuje i uruchamia usługi na każdym poziomie,
- każdemu poziomowi pracy jest przyporządkowany podkatalog */etc/rc<#>.d*,
- nazwa usługi (skryptu) w tych katalogach ma postać: *K##nazwa* lub *S##nazwa*

Zatrzymywanie - etapy I

- wydanie polecenia **shutdown**,
- system powiadamia użytkowników o zatrzymaniu,
- wszystkie procesy użytkowników są zatrzymywane,
- system przechodzi na poziom 0, lub 6,
- zatrzymanie odpowiednich usług,
- odmontowanie systemów plików,
- zatrzymanie lub restart systemu.

Plan punktu: Nowe rozwiązania w Linux v2.6

Nowe rozwiązania w Linux v2.6

Nowe rozwiązania w Linux v2.6 I

- częste użycie *initrd*
- wykorzystywanie *udev*, *hotplug*, *HAL*
- SATA i nazwy urządzeń `hd`, `sd`
- użycie *USB Mass Storage*
- *Upstart* (Ubuntu) zamiast typowego *Init*
- używanie *devicemapper* + LVM (EVMS).

Wykład: Procesy i monitorowanie systemu

Plan wykładu I

Procesy w Unixie

Automatyczne uruchamianie procesów

Rejestrowanie zdarzeń

Motto I

No one gets too old to learn
a new way of being stupid

Plan punktu: Procesy w Unixie

Procesy w Unixie

Programy i procesy I

- program – zestaw danych i instrukcji przechowywanych w postaci pliku,
- program jest „statyczny”,
- proces – środowisko pracy programu istniejące w pamięci systemu; wykonywany program,
- proces jest „dynamiczny”,
- proces jest identyfikowany przez PID (*Process IDentifier*).

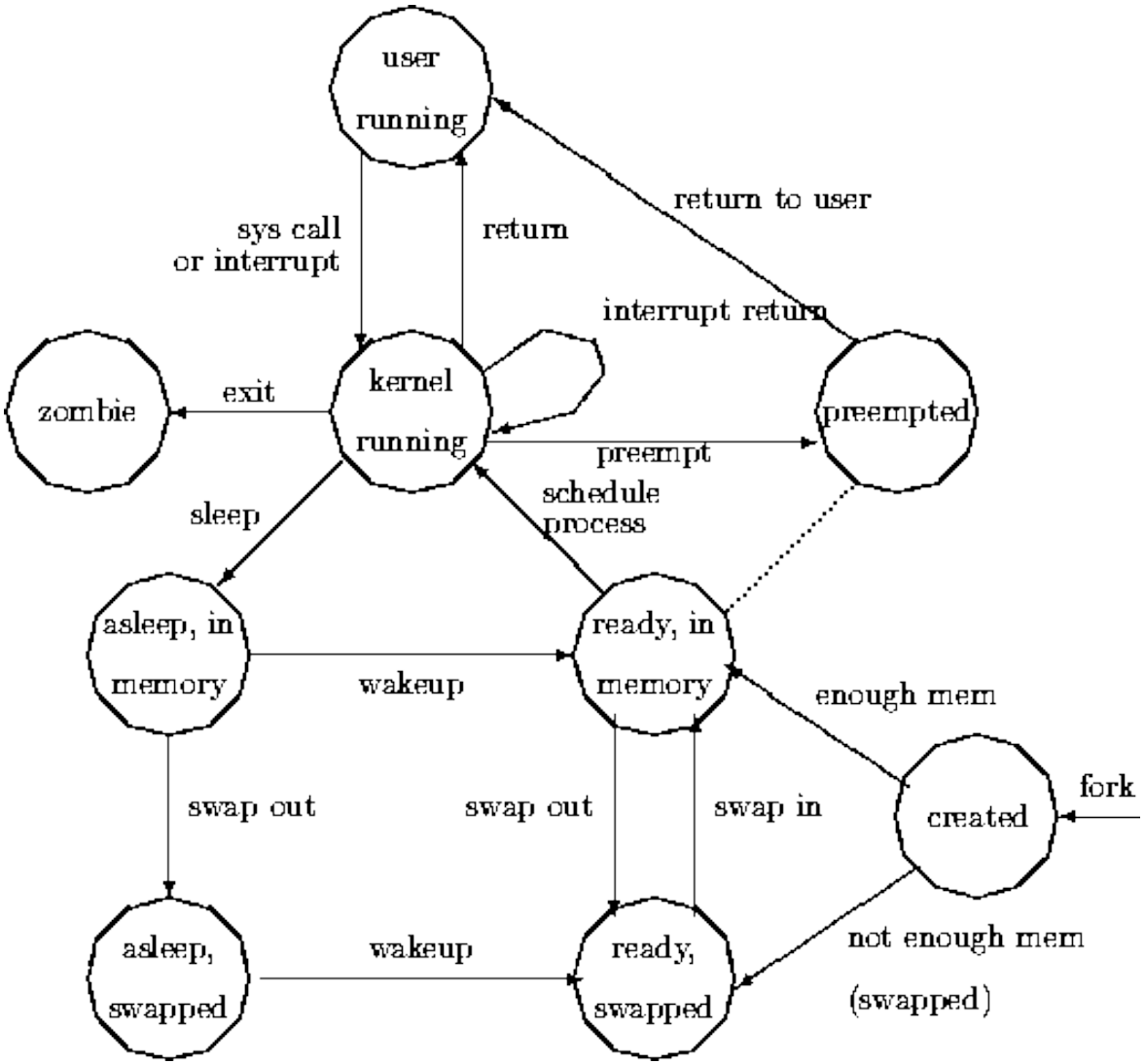
Hierarchia procesów I

- każdy proces jest identyfikowany przez PID (*Process IDentifier*),
- pierwszy proces o (PID=0) działa w obrębie jądra w sposób niejawny (ew. *scheduler*),
- jądro tworzy proces *Init* o PID=1,
- Init tworzy kolejne procesy,
- wszystkie następne procesy mogą być utworzone tylko przez Init lub inne procesy, (procesy potomne – dzieci (*child process*))
- każdy proces ma proces nadrzędny – rodzica (*parent*); rodzica określa *PPID - ParentPID*.

Tworzenie procesów I

1. rodzic tworzy swoją kopię przez `fork()`,
2. kopia, czyli proces potomny, może wykonać `exec()` w celu uruchomienia innego programu,
3. proces potomny może zakończyć działanie przez `exit()`,
4. rodzic może czekać na zakończenie potomka przez `wait()`,
5. przy wykonywaniu asynchronicznym `sleep()`, `kill()`, i inne.

Stany procesu w Unixie I



Właściciele procesu I

Właściciel i grupa procesu jest ustawiany taki, jak właściciel i grupa konta, z którego jest uruchamiany.

```
$ ls -l /usr/bin/mc
-rwxr-xr-x  1 root root /usr/bin/mc
$ /usr/bin/mc
$ ps u
USER  PID   COMMAND
gjn   1972  /usr/bin/mc
```

Programu SU/G/ID I

Wyjątkiem sa programy SU/G/ID w przypadku których odpowiednio właściciel i grupa są brani z pliku programu.

```
$ ls -l /usr/bin/passwd
-rwsr-xr-x  1 root root /usr/bin/passwd
$ /usr/bin/passwd
$ ps u
USER  PID   COMMAND
root  2695  /usr/bin/passwd
```

Priorytety procesów I

- system przydziela procesom czas procesora na podstawie priorytetu, na który wpływa wartość *nice*,
- każdy proces ma własny priorytet,
- domyślna wartość *nice* to 0,
- nice przyjmuje wartości od -20 (większy priorytet) do 19 (mniejszy priorytet),
- użytkownik może zmieniać, zwiększać *nice* tylko swoich procesów,
- tylko administrator może nadawać *nice* ujemny (zwiększać priorytet).

Polecenie nice I

nice

Polecenie `nice` służy do uruchamiania programu z zadanym priorytetem (domyślnie 10).

```
nice -n wart_nice polecenie
```

renice

Do zmiany wartości *nice* działającego procesu służy polecenie `renice`.

```
renice wart_nice PID
```

To polecenie umożliwia również symboliczne podawanie nazw procesów i nazw użytkowników przy pomocy opcji `-p -u`, oraz grup procesów przy pomocy `-g`.

Przykład użycia nice I

```
$ nice bash
$ ps l
 UID   PID  NI   COMMAND
1000   181   0   -bash
1000   846  10    bash
1000   848  10    ps l
$ renice 20 846
846: old priority 10, new priority 20
$ ps l 846
 UID   PID  NI   COMMAND
1000   846  19   bash
```

Polecenie ps I

Jest to podstawowe polecenie umożliwiające wyświetlanie informacji o procesach.

a	wszystkie procesy
x	procesy nie korzystające z terminala
l	format „długi”
u	format związany z użytkownikiem
m	informacje o pamięci
f	drzewo procesów
e	informacje o środowisku
txx	procesy związane z terminalem xx

Tablica: Opcje ps

Przykład użycia ps I

- Wyświetlanie własnych procesów w danej sesji

  ```
  $ ps
  ```

- Wyświetlanie wszystkich procesów związanych z powłokami użytkowników

  ```
  $ ps a
  ```

- Wyświetlanie wszystkich procesów w systemie

  ```
  $ ps ax
  ```

Procesy systemowe I

- Istnieją pewne procesy krytyczne dla działania systemu.
- Są one tworzone bezpośrednio przez jądro systemu, czasami jako wątki.
- Zajmują się np. zarządzaniem pamięcią wirtualną, obsługą zdarzeń sprzętowych, synchronizacją buforów I/O, itp.
- Mają one najczęściej identyfikatory o kolejnych numerach od 2 w górę.
- Mogą zależeć od konfiguracji systemu, czy typu jądra.

Przykład Linux 2.4 I

```
1   init
2   [keventd]
3   [ksoftirqd_CPU0]
4   [kswapd]
5   [bdflush]
6   [kupdated]
174 [kjournald]
314 [kcopyd]
488 [khubd]
```

Przykład Linux 2.6 I

```
1 init
2 [migration/0]
3 [ksoftirqd/0]
10 [events/0]
14 [khelper]
29 [kblockd/0]
105 [pdflush]
108 [aio/0]
107 [kswapd0]
```

Przykład FreeBSD 5 I

```
0 [swapper]
1 init
2 [g_event]
3 [g_up]
4 [g_down]
5 [kqueue taskq]
6 [thread taskq]
7 [acpi_task0]
10 [ktrace]
11 [idle]
12 [irq1]
49 [usb1]
51 [fdc0]
52 [swi0: sio]
53 [pagedaemon]
54 [vmdaemon]
55 [pagezero]
56 [bufdaemon]
58 [syncer]
```

Przykład FreeBSD 5 II

```
64 [schedcpu]
```

Sygnały procesów I

- sygnały są jedną z podstawowych metod komunikacji między procesami,
- umożliwiają specjalną komunikację użytkownika z procesami,
- jądro systemu wysyła sygnały do procesów,
- większość sygnałów jest związanych z różnymi warunkami zakończenia procesów,
- niektóre mogą być przechwytywane,
- nieprzechwytywalnym sygnałem powodującym bezwarunkowe usunięcie procesu jest `SIGKILL` (9).

Polecenie `kill` I

- `kill` przesyła zadane sygnały do procesów
- wywołanie polecenia ma postać:
  `kill -sygnal PID`
- listę sygnałów można uzyskać przy pomocy polecenia: `kill -l`
- użytkownik systemu może przesyłać sygnały tylko do swoich procesów,
- administrator może przesyłać sygnały do wszystkich procesów,
- domyślnym wysyłanym sygnałem jest `SIGTERM` (15).

Przykład użycia kill I

```
$ ps tp6
  PID TTY STAT TIME COMMAND
  712  p6 S    0:00 -bash
$ kill 712
$ ps tp6
  PID TTY STAT TIME COMMAND
  712  p6 S    0:00 -bash
$ kill -9 712
 Done                    xterm -ls
$ ps tp6
  PID TTY STAT TIME COMMAND
No processes available.
```

Lista sygnałów I


```
$ uname ; kill -l
Linux
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     16) SIGSTKFLT
17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU
25) SIGXFSZ     26) SIGVTALRM   27) SIGPROF     28) SIGWINCH
29) SIGIO       30) SIGPWR      31) SIGSYS      34) SIGRTMIN
35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3  38) SIGRTMIN+4
39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12
47) SIGRTMIN+13 48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14
51) SIGRTMAX-13 52) SIGRTMAX-12 53) SIGRTMAX-11 54) SIGRTMAX-10
55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7  58) SIGRTMAX-6
59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

Lista sygnałów II

```
$ uname ; kill -l
SunOS
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGEMT       8) SIGFPE
 9) SIGKILL     10) SIGBUS      11) SIGSEGV     12) SIGSYS
13) SIGPIPE     14) SIGALRM     15) SIGTERM     16) SIGUSR1
17) SIGUSR2     18) SIGCHLD     19) SIGPWR      20) SIGWINCH
21) SIGURG      22) SIGIO       23) SIGSTOP     24) SIGTSTP
25) SIGCONT     26) SIGTTIN     27) SIGTTOU     28) SIGVTALRM
29) SIGPROF     30) SIGXCPU     31) SIGXFSZ     32) SIGWAITING
33) SIGLWP      34) SIGFREEZE   35) SIGTHAW     36) SIGCANCEL
37) SIGLOST     41) SIGRTMIN    42) SIGRTMIN+1  43) SIGRTMIN+2
44) SIGRTMIN+3  45) SIGRTMAX-3  46) SIGRTMAX-2  47) SIGRTMAX-1
48) SIGRTMAX
```

Strumienie I

Każdy proces pracujący w środowisku Unix komunikuje się z otoczeniem m.in. przez strumienie. Domyślnie są otwierane 3:

- 0 *standard input* (*stdin*)
- 1 *standard output* (*stdout*)
- 2 *standard error (output)* (*stderr*)

Przekierowania strumieni I

Powłoka pozwala na przekierowanie strumieni, w **sh** znakami:

- `[n] > plik`
- `[n] >> plik`
- `< plik`

Przykłady:

```
ps > foo
w 2 > bar
ls >foo 2>bar
```

Komunikacja między procesami I

Powłoka pozwala na realizowanie komunikacji między procesami przez *łączenie strumieni*, uruchamiane przez znak | (ang. *pipe*). Powoduje on połącznie *stdout* zadania i z *stdin* zadania $i+1$. Przykład:

```
ls | wc
ps | sort
```

Pozwala to na tworzenie dowolnie złożonych *filtrów*:

```
ps -A | grep tcsh | wc
cut -d ':' -f 5 /etc/passwd
 | grep -i ewa | wc -l
```

Plan punktu: Automatyczne uruchamianie procesów

Automatyczne uruchamianie procesów
- Polecenie at
- System Cron

Automatyczne uruchamianie procesów I

Automatyzacja niektórych prac systemu jest niezwykle pomocna, używa się do niej programów:

- `at` umożliwia jednorazowe uruchomienie programu o zadanym czasie,
- `cron` umożliwia cykliczne uruchamianie programów,
- obydwa systemy mogą być używane przez wszystkich użytkowników,
- składają się z części serwera (demona) i klienta – uruchamianego przez użytkownika,

Cron a At I

- serwer `atd` kolejkuje zadania każdego użytkownika,
- w przypadku `cron` każdy użytkownik ma jeden plik `crontab`, w którym zapisuje wszystkie zadania,
- `cron` udostępnia dodatkową funkcjonalność administratorowi,
- `cron` jest jednym z podstawowych narzędzi automatyzujących pracę administratora.

Polecenie at I

Kontrola dostępu do polecenia odbywa sie przez dopisanie reguł *allow/deny* dla grup użytkowników w plikach `/etc/at.allow`, `/etc/at.deny`. Format wywołania:
`at godzina [data] [opóźnienie] polecenie`
Czas można podawać na różne sposoby, np.: tylko godzina, pełna data, opóźnienie. Wszystkie możliwości znajdują się w podręczniku systemowym man.

Opcje at I

Polecnie `at` ma następujące opcje i odpowiadające im polecenia pomocnicze:

at czas	umieszcza nowe zadanie w kolejce,
at -m czas	wymusza wysłanie email'a po zakończeniu zadania,
at -l (atq)	wyświetla kolejkę zadań,
at -d nr (atrm nr)	usuwa podane zadanie z kolejki,
at -c nr	wyświetla skrypt opisujący zadanie.

Polecenie crontab I

- do komunikacji z demonem `crond` służy polecenie `crontab`,
- `crontab` umożliwia edycję i pokazywanie pliku `crontab`,
- każdy użytkownik ma jeden taki plik,
- w pliku może się znajdować wiele zadań, każde w osobnej linii,
- polecenie `crontab` korzysta z zewnętrznego edytora do edycji pliku.

Opcje crontab I

Podstawowe wywołania `crontab` to:

- `crontab -e` – edycja pliku,
- `crontab -l` – wyświetlenie pliku,
- `crontab -r` – usunięcie pliku,
- `crontab -u user {-e|-l-r}` – edycja pliku zadanego użytkownika – wywołanie tylko dla administratora,
- `crontab plik` – użycie gotowego `pliku` jako `crontab`.

Składnia crontab I

- linia składa się z sześciu pól, ostatnie to nazwa polecenia do wykonania,
- pola w każdej linii oznaczają kolejno: minutę, godzinę, dz. mies., miesiąc, dz. tyg.,
- gwiazdka w miejscu wartości oznacza: „dla każdej wartości pola”, np.: co godzinę,
- kilka wartości rozdziela się przecinkami,
- znak „–” pozwala na podanie zakresu wartości,
- połącznie go ze znakiem „/” umożliwia podanie kroku w obrębie zakresu.

Przykład crontab I

Przykładowy plik crontab:

```
# codziennie o 6 rano
0 6 * * *       who
# o godzinie 14:15 kazdego 28. dnia miesiaca,
15 14 28 * *    ps
# co dwie godziny
0 0-23/2 * * *  finger
```

Plik /etc/crontab I

- plik /etc/crontab zawiera centralną konfigurację demona cron,
- ma możliwość podania przed poleceniem nazwy użytkownika z prawami którego będzie wykonywane polecenie,
- plik ten może mieć taką zawartość:

```
42 6 * * * root run-parts /etc/cron.daily
47 6 * * 7 root run-parts /etc/cron.weekly
52 6 1 * * root run-parts /etc/cron.monthly
```

- polecenie run-parts uruchamia wszystkie programy wykonywalne w podanym katalogu,
- aby zapewnić uruchamianie programu każdego dnia, tygodnia, miesiąca, trzeba go umieścić w odpowiednim katalogu.

Plan punktu: Rejestrowanie zdarzeń

Rejestrowanie zdarzeń
- Przetwarzanie plików rejestrowych
- Rozbudowa systemu rejestrowania

Pliki rejestrowe I

Linux ma mechanizmy rejestrujące każde zdarzenie w systemie.

- rejestrowanie (logowanie; `log` – dziennik) oznacza systematyczne zapisywanie odpowiednich informacji do plików,
- pliki rejestrów określa się po ang. *log files*,
- logowanie jest wykonywane przez pracujący bez przerwy system Sysklogd, składający się z demonów: `klogd` i `syslogd`,

Syslog I

- system Sysklogd zapisuje (loguje) informacje do plików rejestrów/„logów” systemowych, w katalogu `/var/log`,
- pliki rejestrów są plikami tekstowymi o jednolitej składni,
- pliki rejestrów są cyklicznie porządkowane przy pomocy specjalnych programów wywoływanych przez Cron.

Demon `syslogd` I

- ten demon jest jednym z najważniejszych demonów systemowych,
- `syslogd` rejestruje wszystkie informacje o pracy systemu,
- jest uruchamiany jako jeden z pierwszych procesów i zamykany jako jeden z ostatnich,
- konfiguracja `syslogd` znajduje się w pliku `/etc/syslog.conf`

Konfiguracja `syslogd` I

- plik `/etc/syslog.conf` (5) zawiera reguły według których `syslogd` sortuje informacje i zapisuje je do różnych plików rejestrowych,
- reguła to dwa pola: *selector action*,
- *selector* ma 2 części *facility.priority*,
- *facility* określa system z którego pochodzi wiadomość np.: auth, kern, mail, syslog,
- *priority* mówi o stopniu ważności wiadomości, np.: debug, warn, err, emerg.
- w polu *action* może być wpisany plik, urządzenie sterujące terminalem.

Przykład /etc/syslog.conf I

Fragment /etc/syslog.conf:

```
auth,authpriv.* /var/log/auth.log
*.*;auth,authpriv.none /var/log/syslog
cron.*       /var/log/cron.log
daemon.*    -/var/log/daemon.log
kern.*      -/var/log/kern.log
lpr.*       -/var/log/lpr.log
mail.*       /var/log/mail.log
mail.err     /var/log/mail.err
*.*          /var/log/all
*.emerg      /dev/tty10
```

Pliki rejestrowe systemu I

- nazwy i zawartość zależą od konfiguracji `syslogd`,
- pliki rejestrów są plikami tekstowymi,
- składają się z linii o określonym formacie, przeważnie jest to: `data godzina hostname proces: komunikat`
- umieszczane są w katalogu `/var/log`,
- niektóre duże programy jak Apache czy FTP mogą zakładać w `/var/log` własne katalogi z logami.

Plik rejestrowe – przykłady I

/var/log/auth.log:

```
Oct 28 00:59:06 enterprise login[170]: ROOT LOGIN on ‘tty1’
Oct 29 08:38:38 enterprise su: (to root) gjn on /dev/ttyp3
Nov  1 06:42:20 enterprise su: (to www-data) root on none
Nov 24 22:28:26 enterprise su: FAILED SU (to root) gjn on /dev/ttyp5
```

/var/log/daemon.log:

```
Jan  2 15:13:36 enterprise sshd[816]: log: Server listening on port 22.
Jan  2 15:13:36 enterprise sshd[816]: log: Generating 768 bit RSA key.
Jan  2 15:13:37 enterprise sshd[816]: log: RSA key generation complete.
Feb 10 13:10:31 enterprise in.smtpd[287]: connect from localhost
```

/var/log/debug:

```
Oct 16 08:24:20 enterprise kernel: ISO 9660 Extensions: RRIP_1991A
Oct 16 08:25:22 enterprise kernel: VFS: Disk change detected on device ide1(22,0)
```

Rotacja logów I

- przez rotację plików rejestrów rozumie się cykliczne usuwanie starszych fragmentów plików,
- przykład prostego algorytmu rotacji przedstawiony jest dalej,
- w większości dystrybucji istnieją dodatkowe narzędzia ułatwiające rotację logów,
- ponieważ rotacji powinno się dokonywać systematycznie, wykorzystuje się Cron,
- do rotacji logów służy narzędzie `savelog` z pakietu DebianUtils,
- używa się też programu `logrotate` z pakietu LogRotate.

Algorytm rotacji I

Dla każdego rotowanego pliku rejestru (`plik`) tworzy się pliki o nazwach: `plik.0 plik.1 ...plik.n-1 plik.n` . Limit ilości plików (liczba n) podaje się programowi służącemu do rotacji logów. Rotację plików rejestrów można przedstawić w postaci prostego algorytmu:

1. nazwa pliku o najwyższym numerze, `plik.k`, jest zmieniana na `plik.k+1` jeżeli $k < n$, w przeciwnym przypadku plik jest usuwany,
2. dla wszystkich plików o numerach $0 \leq j \leq k-1$ zmienia się nazwy z `plik.j` na `plik.j+1`,
3. aktualny `plik` jest przenoszony do `plik.0`,
4. tworzony jest nowy `plik` o długości 0 bajtów,
5. zmieniane są odpowiednio prawa dostępu do `plik plik.0`.

Rotacja – LogRotate I

1. upewnić się, że zainstalowany jest pakiet LogRotate,
2. dokonać ewentualnej konfiguracji w `/etc/logrotate.conf`,
3. stworzyć plik konfiguracyjny dla rotowanego logu (przykład dalej),
4. skonfigurować Cron tak, by `logrotate` dla tego pliku był odpowiednio często uruchamiany,
5. można to zrobić umieszczając go w katalogu `/etc/logrotate.d`.

Modyfikacje standardowego systemu I

System Syslog jest używany w środowisku systemów Unix od dawna. Jest to narzędzie sprawdzone, lecz posiadające pewne ograniczenia. Poważnym ograniczeniem często okazuje się prosty system sortowania komunikatów. Syslog ma tylko 12 kategorii komunikatów i osiem poziomów ich ważności. Nie ma również wbudowanych mechanizmów sortowania komunikatów na podstawie ich treści – musi być to realizowane przez zewnętrzne programy. Syslog nie posiada również mechanizmów kontroli dostępu przy przesyłaniu komunikatów przez sieć. Zostały podjęte próby napisania narzędzi, które usuwają te ograniczenia.

Syslog-NG I

Zasługującym na zainteresowanie narzędziem jest Syslog-NG (Next Generation). Został napisany w firmie BalaBit Computing i jest dostępny pod adresem `http://www.balabit.com/products/syslog-ng`. Syslog-NG jest lepiej konfigurowalny, umożliwia sortowanie wiadomości na podstawie ich zawartości, ma mechanizmy umożliwiające kontrolę integralności plików rejestrowych oraz ich szyfrowanie, posiada także rozbudowane funkcje przesyłania komunikatów przez sieć. Syslog-NG pracuje na platformach Linux, BSD, Solaris.

Koniec

Czy są jakieś pytania?