Wprowadzenie do systemu Unix

Witold Paluszyński
Katedra Cybernetyki i Robotyki
Politechnika Wrocławska
http://www.kcir.pwr.edu.pl/~witold/

Historia

1966-1969: w Bell Labs (AT&T) powstaje Unix na komputer PDP-7

1 stycznia 1970: 0:00 — godzina zero systemu Unix

lata 70-te: powolny wzrost popularności, głównie w instytucjach badawczych i akademickich

1976 wersja szósta (Sixth Edition) — Univ.of California Berkeley wykupił prawa do kodu systemu Unix i rozpoczął prace nad własną odmianą BSD (Berkeley Software Distribution)

lata 80-te: wersja komercyjna AT&T: System III, V — wdraża się powoli

1983 4.2BSD: pełne oprogramowanie TCP/IP

1984 100,000 instalacji Uniksa na różnych platformach sprzętowych

1988 początki standaryzacji Uniksa: POSIX, później X/Open

lata 90-te: dalszy rozwój Unixa: dojrzały, stabilny system z dobrze rozwiniętą warstwą sieciową, łatwy do przenoszenia na nowe platformy sprzętowe

koniec XX wieku: popularność Linuxa

wiek XXI: rewolucja open-source, powstaje wiele systemów oprogramowania

Standardy

standardy interface'u systemowego Unixa:

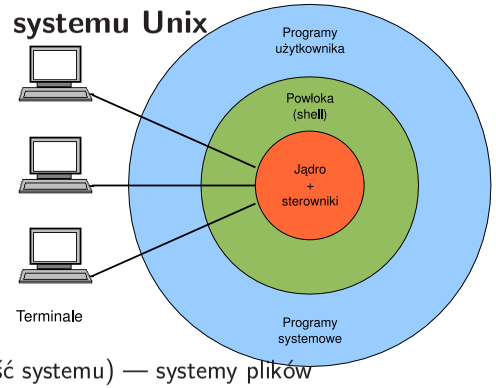
POSIX.1 (Portable Open System Interface) —
IEEE 1003.1 1988/1990, ISO/IEC 9945-1:1990, uzupełnienia:
POSIX.1b-1993 (rozszerzenia czasu rzeczywistego), POSIX.1c-1996 (wątki)

XPG (X/Open Portability Guide) —
konsorcjum X/Open: XPG3 1989, XPG4 1993, uzupełnia standardy POSIX o standard AT&T SVID3 (System V Interface Definition Issue 3)

inne standardy Unixa o mniejszym znaczeniu: standard interpretera komend i aplikacji systemowych IEEE 1003.2 (POSIX.2 - 1992), standard administracji systemu IEEE 1003.7

Architektura systemu Unix

jądro
powłoka
programy
terminale



hardware:

- urządzenia dyskowe (integralna część systemu) — systemy plików
- tzw. interface sieciowy (również pod kontrolą systemu),
- inne urządzenia I/O (terminale, drukarki, modemy, napędy taśm, itp.)

software:

- jądro zarządza sprzętem i umożliwia równoległe lub quasi-równoległe uruchamianie procesów w wirtualnych przestrzeniach adresowych
- drivery urządzeń wkompilowane lub dynamicznie dolinkowane do jądra
- system kompilatora C zawiera pliki nagłówkowe i funkcje systemowe do korzystania z zasobów systemu

Konta użytkowników

- Podstawowe atrybuty: numery i nazwy kont, grupy, hasło, kartoteka dyskowa, desygnowany interpreter komend.
- Inne „rozproszone” atrybuty użytkownika: ulimit, quota, itp., np. uprawnienia do drukowania na konkretnej drukarce.
- Zakładanie i kasowanie kont użytkowników, pliki administracyjne (passwd i shadow).
- Użytkownicy w konfiguracji sieciowej (YP/NIS, NIS+, LDAP).

Uprawnienia użytkowników

- Podstawowe uprawnienia użytkownika chronione hasłem: dostęp do kartoteki i plików własnych (również poza kartoteką własną, np. skrzynka pocztowa), ustawianie praw własności i praw dostępu plików, maska tworzenia plików.
- Nabywanie praw innych użytkowników: su user, su - user, newgrp, pliki setuid, setgid, oddawanie plików innym użytkownikom.
- Mechanizm grup do współdzielenia uprawnień do wybranych plików lub kartotek.
- Mechanizm list ACL do współdzielenia uprawnień do wybranych plików lub kartotek.

Grupy użytkowników

W systemie Unix istnieje koncepcja grup użytkowników. Tworzenie grup użytkowników miało w oryginalnym zamyśle ułatwić zarządzanie uprawnieniami do plików. Każdy użytkownik niejako z obowiązku należy do jakiejś grupy użytkowników, i można tworząc konta użytkowników od razu razem z nimi tworzyć pewne grupy. Gdy taka grupa odpowiada rzeczywistej grupie współpracujących ludzi, to mogą oni łatwo tworzyć, udostępniać sobie nawzajem, i korzystać z plików należących do tej grupy.

Użytkownik mógł tworzyć pliki — i pozostawać ich twórcą i właścicielem — ale grupowym właścicielem plików stawała się systemowa grupa tego użytkownika. Użytkownik/właściciel mógł nadać prawa dostępu do tego pliku (o nadawaniu praw dostępu będzie poniżej) dla grupy, pozwalające na odczyt i/lub zapis do pliku dla całej grupy.

Tworzenie i kasowanie plików jest związane z uprawnieniem typu „write" do katalogu zawierającego dany plik. Zatem, podobnie nadając uprawnienia dla konkretnych katalogów, ich właściciel mógł w efekcie zarządzać uprawnieniem do tworzenia i kasowania plików w danym katalogu.

Problem z grupami: grupy dodatkowe

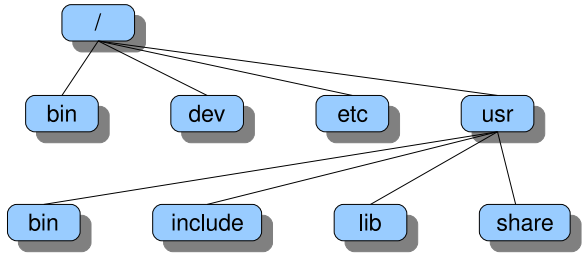
Przydatna na pozór koncepcja grup szybko okazała się niewystarczająca. Na przykład, wykładowca mógłby chcieć stworzyć grupę KursUnix aby udostępniać pliki studentom tego kursu. Co jednak zrobić, jeśli inny wykładowca chciałby w podobnym celu stworzyć grupę SystemyOperacyjne? Do której grupy powinni należeć studenci?

Widać, że koncepcja kiedy jeden użytkownik należy do dokładnie jednej grupy jest niewystarczająca. Dlatego jako rozwiązanie tego problemu istnieje mechanizm grup dodatkowych. Użytkownicy rzeczywiście należą do jednej grupy podstawowej, określonej w pliku `/etc/passwd`, ale mogą należeć do dowolnej liczby innych grup, według specyfikacji zawartej w pliku `/etc/group`.

Korzystanie z uprawnień grup dodatkowych odbywa się w różnych systemach w nieco inny sposób. Większość systemów Unix wymaga użycia w tym celu polecenia `newgrp`, które uruchamia nową instancję interpretera poleceń, z nową grupą. W systemach Linux użytkownik automatycznie posiada uprawnienia wszystkich grup dodatkowych, do których należy.

System plików: katalogi i pliki

Dyskowy system plików jest podstawą działania systemu Unix. Zawiera zarówno katalogi i pliki systemowe, jak i programy i pliki aplikacji, oraz katalogi i pliki użytkowników w jednej hierarchicznej strukutrze.



Dodatkowe dyski zawierające systemy plików mogą być dołączane do systemu w dowolnym miejscu systemu plików operacją **mount**.

Katalogi systemowe

/ — „korzeń" systemu plików

/etc — katalog zawierający najważniejsze pliki konfiguracyjne i startowe

/dev — katalog zawierający tzw. pliki urządzeń stanowiące punkty wejścia do różnych driverów, pseudourządzenia, itp.

/usr — tradycyjnie katalog zawierający oprogramowanie systemowe i użytkowe (historycznie zawierał również katalogi użytkowników, obecnie zwykle w /home)

/bin /usr/bin — programy systemowe i użytkowe

/sbin /usr/sbin — programy administracyjne

/var — pliki o zmiennej zawartości, głównie rejestry (logi) systemowe, pliki robocze podsystemu poczty, podsystemu drukowania, itd.

/lib /usr/lib /usr/X11/lib ... — katalogi zawierające biblioteki procedur binarnych

/usr/include — pliki nagłówkowe kompilatora C

/usr/local — oprogramowanie doinstalowywane do systemu: tworzone lokalnie i komercyjne

/tmp — pliki tymczasowe, często umieszczony na RAM-dysku, okresowo automatycznie czyszczony

/home — zbiór katalogów użytkowników

Pliki systemowe

/etc/passwd

```
root:x:0:1:Super-User:/:/sbin/sh
powerdown:x:0:1:Power Down User:/:/usr/local/sbin/powerdown
reboot:x:0:1:Reboot User:/:/usr/sbin/reboot
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
sys:x:3:3::/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
witold:x:101:100:Witold Paluszynski,p.307/C-3:/home/witold:/usr/bin/tcsh
```

/etc/group

```
root::0:root
other::1:
bin::2:root,bin,daemon
sys::3:root,bin,sys,adm
adm::4:root,adm,daemon
users::100:
```


System plików: różne typy plików

Pliki: zwykłe pliki (ciągi bajtów), pliki specjalne (znaki specjalne w nazwach plików `*,?,[],{}` interpretowane przez interpreter poleceń):

```
bash-3.00$ ls -lR /etc
/etc:
total 619
lrwxrwxrwx   1 root     root          14 kwi  5  2005 TIMEZONE -> ./default/init
drwxr-xr-x   6 root     other        512 kwi  5  2005 X11
...


kartoteka:
drwxrwxr-x  27 root     sys         3584 Oct 16 15:17 /etc

zwykle pliki: plik tekstowy, plik wykonywalny:
-r--r--r--   1 root     sys         1129 Sep 29 11:05 /etc/passwd
-r-xr-xr-x   1 bin      bin        13872 Oct 25  1995 /usr/bin/ls
```


System plików: pliki specjalne

Pliki specjalne umożliwiają dostęp do rzeczywistych urządzeń pod kontrolą systemu (pliki typu “c” i “b”), albo do unixowych mechanizmów wejścia/wyjścia takich jak potoki (pliki typu “p”) lub gniazdka (pliki typu “s”):

```
pliki specjalne urzadzen:
crw--w----   1 uucp     tty      106,  1 Oct  4 05:54 /dev/tty01
brw-rw-rw-   1 root     sys       36,  2 Jul 10  1997 /dev/fd0

"nazwany potok" (named pipe, FIFO):
prw-rw-rw-   1 root     root          0 Oct 16 15:17 /tmp/.asppp.fifo

gniazdko:
srwx------   1 witold   gurus         0 Oct 20 18:25 /tmp/jpsock.150_01.1838=
```

System plików: dowiązania (linki)

- Dowiązania plików (linki): różnych pozycje w katalogach odwołujące się do tego samego pliku.
- Linki mogą mieć różne nazwy, lecz zarówno zawartość pliku jak i wszystkie atrybuty są identyczne.
- Po utworzeniu drugiego linku do pliku nie można odróżnić który był oryginalny; stąd wszystkie pozycje plików w katalogach nazywa się linkami.
- Dodatkowe linki do pliku mogą istnieć tylko w ramach tej samej struktury dyskowej.

```
jeden plik z wieloma dowiązaniami (linkami):
       265 -r-xr-xr-x   5 root     bin       240264 sty 23  2005 /usr/bin/edit
       265 -r-xr-xr-x   5 root     bin       240264 sty 23  2005 /usr/bin/ex
       265 -r-xr-xr-x   5 root     bin       240264 sty 23  2005 /usr/bin/vedit
       265 -r-xr-xr-x   5 root     bin       240264 sty 23  2005 /usr/bin/vi
       265 -r-xr-xr-x   5 root     bin       240264 sty 23  2005 /usr/bin/view
```


System plików: linki symboliczne

- Linki symboliczne: pliki specjalne zawierające odwołania do innych plików; pełnią podobną funkcję jak linki, lecz są inaczej skonstruowane.
- Większość operacji na plikach „widzi" linki symboliczne zupełnie tak samo jak samo jak prawdziwy link do pliku, są jednak operacje specjalne działające tylko na linkach symbolicznych.
- Linki symboliczne mogą istnieć do plików w innych strukturach dyskowych, i mogą również istnieć do nieistniejących plików.
- Linki symboliczne mogą odwoływać się zarówno do plików jak i do katalogów.

```
linki symboliczne:
lrwxrwxrwx   1 root     root          12 Jul 10  1997 /etc/hosts -> ./inet/hosts
lrwxrwxrwx   1 root     root          14 Jul 10  1997 /etc/log -> ../var/adm/log
```


System plików: atrybuty plików i struktura i-node

Atrybuty plików: właściciel i grupa pliku, prawa dostępu, bit set-uid, bit set-gid, sticky bit (pliki: save text, katalogi: /tmp), specjalne kombinacje bitów praw dostępu (np. mandatory record locking = g+s,g-x).

I-node: struktura istniejąca dla każdego pliku zawierająca szereg informacji o tym pliku (w sensie obiektu istniejącego na dysku, a nie linku):

- typ pliku: zwykły, specjalny (urządzenie), kartoteka
- 9 bitów praw dostępu i 3 bity dodatkowe
- długość pliku w bajtach
- numer właściciela
- numer grupy
- czas ostatniej modyfikacji pliku
- czas ostatniej modyfikacji i-node'u
- czas ostatniego dostępu do pliku
- liczba odwołań z różnych kartotek systemu plików, tzw. *linków*
- inne informacje, mniej istotne dla administratora

System plików: 9 bitów praw dostępu

Pierwsze trzy bity 9-bitowego wektora praw dostępu określają prawa właściciela pliku, środkowe trzy bity określają prawa grupowego właściciela pliku, a ostatnie trzy bity — prawa pozostałych użytkowników.

W przypadku katalogów, „r" określa prawo do czytania listy plików katalogu, bez dostępu do tych plików, nawet gdy ich indywidualne prawa taki dostęp dają. Prawo „x" pozwala na przyłączanie się do katalogu i dostępu do zawartych w nim plików, zgodnie z ich własnymi prawami dostępu, ale bez możliwości odczytania listy tych plików. Natomiast „w" określa prawo do tworzenia i usuwania plików w katalogu. Prawo „w" nie ma wpływu na możliwość edycji istniejących plików „w miejscu", ale ma na możliwość zmiany ich nazwy.

Uwaga: semantyka określania praw dostępu mówi, że jeśli UID procesu jest właścicielem pliku, to prawa dostępu są określone przez pierwsze trzy bity, a jeśli UID nie jest właścicielem pliku, ale GID procesu jest grupowym właścicielem pliku, to prawa dostępu są określone przez środkowe trzy bity, NIEZALEŻNIE od praw dostępu określonych przez ostatnie trzy bity. Oznacza to, że prawa dostępu mogą zabronić właścicielowi dostępu do pliku, nawet gdyby jego grupa by na to pozwalała, oraz, że mogą zabronić grupowemu właścicielowi pliku dostępu, gdy pozostali użytkownicy taki dostęp mogą uzyskać.

System plików: 3 dodatkowe bity

```
plik wykonywalny set-uid:
-r-s--x--x   1 root     sys       296300 Sep 22  1997 /usr/bin/admintool

plik wykonywalny set-gid:
-r-xr-sr-x   1 bin      tty         9024 Oct 25  1995 /usr/sbin/wall

plik wykonywalny save-text:
-rwxr-xr-t   1 witold   users    7736776 Sep 30  1996 /usr/local/bin/emacs-19.34

kartoteka ze "sticky bit":
drwxrwxrwt   4 sys      sys          512 Oct 17 06:29 /tmp

zwykly plik z mandatory record locking:
-rwxr-lr-x   2 witold   users     152673 May 30  2000 lp
```


Eksploracja systemu plików: find

```
# znajdowanie plików typu JPEG
find ~ -name '*.jpg' -print

# znajdowanie niedawno modyfikowanych plików (< 10 dni)
find ~ -mtime -10 -print

# to samo, ale chcemy zobaczyć informacje ls o tych plikach
find ~ -mtime -10 -ls

# znajdowanie plików, które nie byly czytane więcej niz 100 dni
find ~ -atime +100 -print
# zauwazmy, ze dla kartotek operacja find modyfikuje daty odczytu

# chcemy zobaczyć daty dostępu, ale:
find ~ -atime +100 -ls
# pokazuje nam domyślnie daty modyfikacji plików

# zeby zobaczyć daty dostępu jawnie, wywolujemy polecenie ls
find ~ -atime +100 -exec ls -lu {} \;

# znajdowanie plików wedlug numeru i-node
```



```
find /etc -inum xxx -print

# inne wywolania
find ~ size +1000000 -ls

find ~ \( -name '*.jpg' -o -name '*.jpeg' -o -name '*.JPG' \) -print

find /etc -type s -ls

find /etc \! -type f -ls

# przyklad z mana:
find $HOME \( -name a.out -o -name '*.o' \) \ -atime +7 -exec rm {} \;
```

Przydatne mechanizmy z find:

find /home -name '*.txt' -print 2>/dev/null

xargs

Terminale

Terminale są artefaktem pierwotnej organizacji systemu komputerowego, kiedy użytkownicy podłączali się do komputerów z rzeczywistych terminali alfanumerycznych przez asynchroniczne łącza szeregowe. Obecnie takie konfiguracje są nadal czasami stosowane, chociaż najczęściej rolę terminala pełni komputer PC z emulatorem terminala znakowego, a zamiast łączy szeregowych wykorzystuje się połączenia sieciowe.

Jednak pojęcie terminala zostało tak głęboko wkomponowane w architekturę systemu Unix, że terminale istnieją i są widoczne nawet pomimo, iż nie wykorzystuje się ich do pierwotnych funkcji. Również system Linux, pomimo iż napisany od nowa dużo później niż Unix, zachowuje tę strukturę.

Rolą terminala jest obsługa znakowego interfejsu użytkownika. Przez terminal użytkownicy mogą włączać się do systemu, i następnie wykonywać różne prace korzystając ze znakowego interpretera poleceń. Ponieważ rzadko kiedy wykorzystuje się w tym celu łącza szeregowe, w systemie istnieją **pseudoterminale**, symulujące transmisję szeregową do i od systemu.

Sterownik terminala

Część jądra Unixa odpowiedzialna za komunikację znakową z terminalami nazywa się sterownikiem terminala (*terminal driver*). Jest to podsystem o wielu parametrach konfiguracyjnych. Jedną z jego ważniejszych funkcji jest buforowanie i edycja wiersza danych, oraz funkcja echa. Buforowanie i prosta edycja wiersza danych z terminala, zapewniana przez sterownik terminala, nazywana jest trybem „ugotowanym" (*cooked*). Funkcje te można wyłączyć, przechodząc do trybu „surowego" (*raw*, albo inaczej: kanonicznego):

```
stty -a
stty -echo
set +o emacs +o vi         # konieczne w bashu
stty echo
stty eof ^a erase ^e kill ^k werase ^w rprnt ^p
stty -icanon
cat > /tmp/proba
ala ma kata.^H^H^H^Hota.
^D
^C
stty icanon
stty sane                  # przydatne kiedy sprawy zajdą za daleko
```


Terminale: bash i readline

Rozbudowany interpreter poleceń `bash` używa mechanizmów, które przeprogramowują sterownik terminala, tworząc funkcjonalność dalece zmieniającą sposób interakcji użytkownika z systemem. Dzieje się tak za pośrednictwem funkcji `readline`, która wczytuje i buforuje wiersz danych użytkownika, pozwalając na jego edycję poleceniami podobnymi do edytora Emacsa lub `vi`.

Funkcjonalność `readline` można włączyć ustawiając opcje basha jednym z poleceń (aczkolwiek jest ona typowo domyślnie włączona):

```
set -o emacs
set -o vi
```

`readline` ma wiele dostępnych funkcji i jest w dużym stopniu konfigurowalna. Polecenia konfiguracyjne można umieszczać w pliku `$HOME/.inputrc`

Terminale: termcap/terminfo

Poza samym przekazywaniem znaków z klawiatury terminala do programu, i z programu na ekran terminala, terminale realizują dodatkowe funkcje. Np. okno terminala może służyć jako wizualny interfejs dla wielu programów, takich jak edytor vi, program top, itp. Rzeczywiste sprzętowe terminale, jak również programowe emulatory terminali, realizują szereg funkcji związanych z wyświetlaniem znaków, jak np. adresowanie kursora, czyli ustawianie kursora znakowego w dowolnej pozycji ekranu. Ponieważ istnieje wiele typów terminali różniących się szczegółami realizacji tych operacji, Unix posiada bazę danych terminali, zwaną `terminfo` albo `termcap`, szczegółowo opisujących te funkcje.

```
echo $TERM;      infocmp
tput clear
tput cup 10 10
tput reset                 # zapamiętaj te dwa polecenia
tput init
tput bel;        tput flash
tput smul
tput rmul
tput rev;        tput blink
tput bold
tput sgr0                  # wyłącza atrybuty znaków
```