Administracja siecią komputerową

Bash, AWK i administracja użytkownikami

Marcin Bieńkowski

Instytut Informatyki
Uniwersytet Wrocławski

Zawartość tego wykładu

- Krótkie wprowadzenie do powłoki Bash
- i jeszcze krótsze do języka AWK
- Użytkownicy i grupy
- Prawa do plików
- PAM
- Sudo

Najprostszy skrypt powłoki

```
echo Hello world!
```

To samo, ale lepiej:

```
#!/bin/bash
ZMIENNA="Hello world!"
echo $ZMIENNA
```


Przekierowania

```
echo "aaa" > plik.txt

PLIK=plik.txt
echo "bbb" >> $PLIK

cat plik.txt > plik2.txt

grep -i tekst -r * 2>/dev/null

ls -al | sed -e "s/[eaoi]/u/g"
```

Parametry dla programu

```
#!/bin/bash

echo pierwszy parametr $1
echo "drugi parametr $2"
echo 'trzeci parametr $3' # niespodzianka!

echo liczba przekazanych parametrow: $#
echo nazwa programu $0
```


Instrukcje warunkowe

```
if [ $# -lt 2]
then
  echo program potrzebuje co najmniej 2 parametrow
fi
```

Albo też:

```
if [[ $# < 2 ]]; then echo program ...; fi
```


Instrukcje warunkowe, ciąg dalszy

```
if grep -q slowo plik
then
  echo plik zawiera slowo
  echo to dobrze
else
  echo plik nie zawiera slowa
fi
```


Operacje arytmetyczne

```
a=5
b=$a+3    # b = "5+3"
let c=$a+3
(( c = $a + 3 ))
d=$(( $a + 3 ))
e=`expr $a + 3`
```

Na marginesie: przechowywanie wyników działania programów

```
ZMIENNA=`program argumenty`
ZMIENNA=`ls`
ZMIENNA=$(ls -al)
```

Pętle

```
#!/bin/bash
licznik=0
while [ ! -z $1 ]
do
  licznik=$((licznik+1))  # można ((licznik++))
  echo Parametr $licznik = $1
  shift
done
```


Pętle, cd.

```
#!/bin/bash
for i in aaa "bbb ccc" "eee" "fff"
do
  echo $i
done
```

wypisze:

```
aaa
bbb ccc
eee
fff
```


Pętle, cd.

Zamiast listy można stosować wywołanie dowolnego programu:

```
for i in $(ls)
do
  echo plik: $i
done
```

Zabicie programów o identyfikatorach od 301 do 350:

```
#!/bin/bash
for i in `seq 50`
do
  kill $((i+300))
done
```


Wprowadzenie do AWK

Program w AWK jest ciągiem par `warunek { akcja }`.
AWK czyta plik wejściowy wiersz po wierszu i dzieli każdy wiersz (na podstawie separatora `FS`) na pola.

Przykład programu:

```
/ala/    { print $1 }
/alaala/ { print $0 }
```

Domyślny warunek: `//`
Domyślna instrukcja `{ print $0 }`

Zmiana separatora FS

Wypisanie wszystkich użytkowników:

```
cat /etc/passwd | awk -F: '{ print $1 }'
```

Wypisanie wszystkich użytkowników z powłoką bash:

```
awk -F: '/\/bin\/bash$/ { print $1 }' /etc/passwd
awk -F: '$7 ~ /bash$/   { print $1 }' /etc/passwd
awk -F "" '{ print $3 }' plik
```


Zmienne

Bloki `BEGIN` i `END` pasują odpowiednio na początku i końcu pliku:

```
BEGIN { x = 0 }
{ x++; print x, $0 }
END { total = print x }
```

Można łatwiej:

```
{ print NR, $0 }
```


Instrukcje sterujące

Można stosować większość znanych z C instrukcji:

```
for (<inicjalizacja>; <warunek podtrzymujący>; <krok>)
if (<warunek>) { ... }
if (<warunek>) { ... } else { ... }
while (<warunek>) { ... }
do { ... } while (<warunek>)
```


Operacje na tekstach

Zamiana podciągu na inny: `sub (co, naco, tekst)`. Wszystkich podciągów: `gsub`

```
x = "abcabc"; sub ("ab", "d", x)
print x   # wypisze dcabc
x = "abcabc"; gsub ("ab", "d", x)
print x   # wypisze dcdc
```

Można stosować wyrażenia regularne

```
x = "XabbXbbbbXa"; gsub (/[ab]+/, "Y&Z", x);
print x;   # wypisze XYabbZXYbbbbZXYaZ
```

Tablice

Tablice:

```
#!/usr/bin/awk

BEGIN { FS = ":" }
{ A[$1] = $3 }
END { for (i in A) { print i, A[i] } }
```


/etc/passwd

Tradycyjnie dane o użytkownikach przechowuje się w pliku `/etc/passwd`

```
root:x:0:0:root:/root:/bin/bash
news:x:9:9:news:/var/spool/news:/bin/sh
bind:x:107:109::/var/cache/bind:/bin/false
mbi:x:1000:100:informacje:/home/mbi:/bin/bash
```

gdzie `informacje = nazwisko,room,tel1,tel2,opis`

W starszych dystrybucjach Linuksa w miejscu `x` było zaszyfrowane hasło. Obecnie stosuje się plik `/etc/shadow`.

/etc/shadow

Plik `/etc/shadow` wygląda następująco:

```
root:aafBdtm8WoKnU:11971:0:99999:7:::
mbi:$1$XqpZQsfO$B3n05YBYEoYAup0Icnd6g/:
    11969:0:99999:7:::
```

Hasła:

- Standardy: DES, MD5
- Salt
- Blokada przez wprowadzenie znaku `*` lub `!` (nie blokuje wszystkiego)

Kolejne pola określają kiedy wygasają hasła. Na przykład 3 pole to liczba dni od 1970 roku, kiedy hasło zostało zmienione po raz ostatni. Ustawienie jej na 0 (`passwd -e user`) wymusza zmianę hasła przy następnym logowaniu.

Zmiana użytkownika

Żeby zalogować się jako użytkownik `newuser`, można

- Zalogować się na wolnej wirtualnej konsoli: `Alt-F1`, `Alt-F2`, ...
- Będąc już zalogowanym – wydać polecenie `su [-] newuser`

Grupy użytkowników

Poza domyślną grupą podaną w pliku `/etc/passwd`

```
mbi:x:1000:100:informacje:/home/mbi:/bin/bash
```

użytkownik może też należeć do dowolnej liczby innych grup.

Plik `/etc/group`:

```
root:x:0:mbi
adm:x:4:mbi
tty:x:5:
dialout:x:20:
cdrom:x:24:mbi,ewa
floppy:x:25:mbi,ewa
audio:x:29:mbi
users:x:100:
```

Użytkownika do grupy można dodać poleceniem `usermod`.

Edycja użytkowników

1. Bardzo niskopoziomowe: ręczna edycja `/etc/passwd`, `/etc/shadow`, `/etc/groups`, tworzenie katalogu użytkownika, jego podstawowa konfiguracja
2. Niskopoziomowe: podstawowe systemowe narzędzia zarządzania powyższymi plikami: `useradd`, `userdel`, `chfn`, `chsh`, `groupadd`, `passwd`...
3. Wysokopoziomowe: narzędzie `adduser`, `deluser`
4. Wysokopoziomowe graficzne narzędzia: np. Webmin, User Manager (Redhat/Fedora), YaST (Suse), Userdrake (Mandriva)

Prawa do plików i katalogów

Z każdym plikiem związane są prawa dostępu. Przykładowo (po wykonaniu `ls -l`):

```
-rw-------   1  mbi   users  /home/mbi/plik.txt
-rwxr-xr-x   1  root  root   /bin/bash
-rwsr-xr-x   1  root  root   /usr/bin/passwd
drwxrwxrwt  10  root  root   /tmp/
drwxrws---   2  mbi   audio  /home/mbi/wspolne/
```

- Polecenia zmiany właściciela lub grupy pliku: `chown`, `chgrp`
- Polecenie zmiany praw dostępu: `chmod`

Linux-PAM

Linux-PAM = Pluggable Authentication Modules for Linux, system uwierzytelniania użytkowników.

- Można go konfigurować osobno dla danej usługi (usługa musi chcieć z niego korzystać) np. `/etc/pam.d/login`, `/etc/pam.d/xscreensaver`
- Zalety: Wspólny system uwierzytelniania dla wszystkich aplikacji, ale jednocześnie można korzystać z różnych metod uwierzytelniania (również innych niż hasła)
- Są 4 rzeczy, które potrafi PAM
  1. `auth` - uwierzytelnianie użytkownika (czy jest tym za kogo się podaje)
  2. `account` - czy ma prawo do danej usługi (ważne konto)
  3. `session` - rzeczy które przy dzieją się przy logowaniu i wylogowywaniu
  4. `password` - mechanizmy zmianie hasła

Linux-PAM
Przykład dla programu login

```
auth requisite pam_securetty.so
auth requisite pam_nologin.so
auth required pam_unix.so

session required pam_env.so readenv=1
session required pam_unix.so
session optional pam_motd.so

account required pam_unix.so

password required pam_unix.so nullok min=4 max=8 md5
```


Program sudo

Umożliwia wykonywanie niektórych lub wszystkich poleceń z uprawnieniami innego użytkownika (lub administratora).

Dlaczego lepsze niż dawanie komuś hasła roota?

- można przydzielać uprawnienia różnym osobom i je potem zabierać
- można określać prawa tylko do niektórych poleceń . . .
- . . . lub/i o określonych porach
- wykonywane polecenia są rejestrowane (do `/var/log/auth.log`)

Program sudo, cd.

Składnia wiersza w pliku `/etc/sudoers`:

```
user komputer=(users) polecenia
```

Przykłady:

```
mbi ALL = (ALL) ALL
mbi valis = (ftp,mail) /bin/ls, /bin/kill
mbi ALL = NOPASSWD: ALL
```

Opcje:

```
Defaults:mbi timestamp_timeout=3
mbi ALL = (ALL) ALL
```


Literatura

- Mendel Cooper
  Advanced Bash-Scripting Guide
  http://www.tldp.org/LDP/abs/html/
- Michael Brennan
  The GNU Awk User's Guide
  http://www.gnu.org/software/gawk/manual/gawk.html
- Lars Wirzenius
  The Linux System Administrator's Guide
  http://www.tldp.org/LDP/sag/html/
- Andrew G. Morgan
  The Linux-PAM System Administrators' Guide
  http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam.html