

Командни процедури во UNIX (прв дел)

Оперативни системи
Аудиториска вежба 3



Вовед

- Долги и комплицирани команди на командна линија.
- Команди кои често се повторуваат.
- Со помош на скрипти (командни процедури) се автоматизира работата во shell околина и се поедноставуваат задачите.
- Задача: Компјутерот да ја заврши задачата на програмерот.

Пишување командни процедури

- Сите shell околина поседуваат вградени функции.
- SH и BASH се најприсутни околина.
- Специфицирање во која околина ќе се извршува командната процедура (прва наредба):

```
#!/path/to/shell (#!/bin/bash)
```

- Се пишуваат во било кој уредувач на текст (pico, kedit, emacs, vi, kwrite...).
- Се снимаат со наставка .sh (skripta.sh).
- По снимањето, се менуваат привилегиите:

```
$ chmod +x skripta.sh
```

- За активирање:

```
$ ./skripta.sh
```

Коментари

- Цел: поедноставно документирање на командните процедури и задачите што ги извршуваат.
- Се користи знакот #.
- Пример

```
#!/bin/bash  
echo "Zdravo, $USER. Ke gi prelistam tvoite datoteki"  
echo "Listam datoteki vo tekovniot imenik, $PWD"  
ls # komanda za listanje na datoteki
```

Променливи

- Како и во сите програмски јазици, командните процедури се незамисливи без променливи.
- Има само еден тип: низа од знаци - STRING.
- Декларирање – не постои.
- Доделување вредност:
 - `imepromenliva=vrednost (X="Zdravo")`
- Добивање вредност на променлива:
 - `$imepromenliva ($X)`
- Да се внимава на празни места:
 - `X = "Zdravo"` – ќе јави грешка
- Доколку вредноста на променливата е празна низа, или, пак е составена од 2 или повеќе зборови, задолжително се користат двојни наводници (""):
 - `X=hello world # error`
 - `X="hello world" # OK`

Пример

```
#!/bin/bash
# dodeli vrednost:
a="hello world"
# prikazi vrednost na "a" :
echo "A e:"
echo $a
```

Излез:

```
A e:
hello world
```

Користење на полиња

- Декларирање:
 - `ime[index]=vrednost`
 - Пример:
 - `FRUIT[0]=apple`
 - `FRUIT[1]=banana`
 - `FRUIT[2]=orange`
 - Пример:
 - `FRUIT=apple`
 - `FRUIT[1]=peach`
 - `name=(value1 value2 ... valueN)`
 - `banda=(jovan miki saso dragana)`

Користење на полиња

- Пристап до еден елемент:
 - `${name[index]}`
 - `echo ${FRUIT[2]}`
- Пристап до сите елементи:
 - `${name[*]}`, `${name[@]}`
 - `echo ${FRUIT[*]}`
- Број на елементи:
 - `${#name[*]}`, `${#name[@]}`
 - `echo ${#FRUIT[*]}`

Read-only променливи и отстранување на променливи

- Read-only променливи

```
FRUIT=kiwi  
readonly FRUIT  
FRUIT=cantaloupe  
echo $FRUIT
```

Излез
kiwi

- Отстранување променливи (unset)

```
unset FRUIT
```

Типови променливи

- Локални променливи
- „Глобални“ променливи
 - `export name=value`
 - `export FMHOME=/usr/frame`
 - `export CLEARHOME=/usr/atria`
- Shell променливи
 - PWD, UID, RANDOM, SECONDS, IFS (“\t\n ”), PATH, HOME

Заштита на променливите со загради ({ })

- За прикажување на вредност на променлива следена со други знаци.

```
#!/bin/bash  
X=ABC  
echo "$Xabc"
```

Грешка

```
#!/bin/bash  
X=ABC  
echo "${X}abc"
```

Правилно

```
#!/bin/bash  
num=2  
echo "Ova e $numra"
```

Грешка

```
#!/bin/bash  
num=2  
echo "Ova e ${num}ra"
```

Правилно

Користење на backslash (\)

- Се користи за приказ на некои од метазнаците:

*** ? [] ' " \ \$; & () | ^ < > newline space tab**

Пример:

```
echo Zdravo; Dobri ste?
```

– Грешка!

```
echo Zdravo\; Dobri ste\?
```

– Ќе се испечати: Zdravo; Dobri ste?

Единечни (' ') и двојни (" ") наводници

- Пред извршување на зададените команди, shell околината се обидува џокер знаците и променливите да ги замени со соодветните вредности.
- На пример, * се заменува со соодветните имиња на датотеки, а променливите се заменуваат со нивните вредности.
- Наводниците го менуваат ваквото однесување.

Пример

- Нека во тековниот именик постојат повеќе датотеки од кои три се со наставки .jpg: glass.jpg, gnome.jpg. и ubuntu.jpg

```
#!/bin/sh  
echo *.jpg
```

Излез:

```
glass.jpg gnome.jpg  
ubuntu.jpg
```

```
#!/bin/sh  
echo "*.jpg"  
echo '*.jpg'
```

Излез:

```
*.jpg  
*.jpg
```

- Двојните наводници
 - спречуваат замена на цокер знаци;
- Единечни наводници се построги:
 - спречуваат замена на цокер знаци;
 - спречуваат замена на променливи.

Примери

```
#!/bin/sh
echo $SHELL
echo "$SHELL"
echo '$SHELL'
```

Излез:

```
/bin/bash
/bin/bash
$SHELL
```

```
#!/bin/sh
echo \*.jpg
echo \$SHELL
```

Излез:

```
*.jpg
$SHELL
```

```
#!/bin/bash
echo -n '$USER='
echo "$USER"
echo "\$USER=$USER"
```

Излез:

```
$USER=milos
$USER=milos
```

```
#!/bin/bash
LS="ls"
LS_FLAGS="-al"
$LS $LS_FLAGS $HOME
```

Излез:

детален приказ на содржината на домашниот именик на нажавениот корисник

Аргументи на командна линија

- Специјални променливи во рамките на командната процедура.
- Се наведуваат по името на командната процедура при нејзиното активирање (одвоени со празни места).
- Пристап: **\$0**, **\$1**, . . . , **\$9**.
- **\$0** се однесува на името на самата командна процедура.
- **\$1** е првиот аргумент, **\$2** е вториот аргумент, итн.
- Доколку има потреба да се референцираат повеќе од 9 аргументи, бројот на аргументот се поставува во загради - **\${nn}**.

Специјални променливи

- Покрај аргументите на командна линија, може да се користат уште неколку специјални променливи:
 - \$#** → го претставува бројот на аргументите на командна линија. Корисен е за контрола на јамки што треба да ги обработат сите аргументи.
- **\$@** или **\$*** → на негово место се заменуваат сите аргументи наведени на командна линија одвоени со празни места. Се користи за праќање на сите аргументи на некоја друга функција во рамките на процедурата или на друга програма.
- **\$\$** → на негово место се заменува id бројот на процесот од shell околината искористена за активирање на командната процедура.

Замена на команди

- Излезот од извршување на командата да се замени на местото на нејзиното име.
- Два начина:
 - `$(command)`
 - ``command``
- Во вториот начин, командата е заградена со спротивни коси наводници – back quotes или backtick (не се обични единечни наводници).
- Во овој случај командата најпрво се извршува во посебна sub-shell околина, по што излезот од командата во командната процедура се заменува на местото на нејзиното име.

```
#!/bin/bash  
my_files=`ls /home/milos`  
echo $my_files
```

Аритметички изрази

- Решавањето на аритметички изрази, исто така е можно на два начини:

Прв начин – `$((expression))`

```
#!/bin/sh
```

```
echo $((1 + 3 + 4))
```

Излез: 8

- Втор начин – `expr op1 operator1 op2 operator2 ...`

```
#!/bin/sh
```

```
echo `expr 1 + 3 + 4`
```

Излез: 8

Структурата if/then/elif/else

- Донесување одлуки во зависност од одредени услови.
- За тестирање на условите и задавање на соодветните наредби, се користи наредбата **if**.

```
if list1 ; then list2 ; elif list3 ;  
then list4 ; else list5 ; fi ;
```

```
if uslov1  
then  
    izraz1  
    izraz2  
    ...  
elif uslov2  
then  
    izraz3  
    izraz4  
    ...  
elif uslov3  
then  
    izraz5  
    izraz6  
    ...  
else izraz7  
fi
```

Структурата if/then/elif/else

- На крајот од извршувањето секоја Unix команда завршува со т.н. излезен статус (exit status). Излезниот статус е всушност цел број, кој укажува дали командата се извршила успешно (тогаш излезниот статус е 0) или неуспешно (има друг излезен статус: различен број - различен проблем).
- За проверка на условите при пишување командни процедури се користи наредбата **test** (се означува со []).
- Доколку условот е точен, оваа команда враќа излезен статус 0. Тоа значи дека наместо командата **test** може да се стави која било друга команда. Доколку командата заврши успешно (излезен статус 0), условот е исполнет и се извршуваат соодветните изрази.

Командата **test** и содветните оператори

- Командата **test** се користи за проверка на условите во структурите за избор. Командата **test** враќа вредност точно (true) или неточно (false), односно враќа излезен статус 0 или различен од нула во зависност од тоа дали условот е исполнет или не е.
 - `test operand1 operator operand2`
- Вообичаено се користи како:
 - `[operand1 operator operand2]`
- Треба да се внимава на празните места околу заградите и меѓу операторот и операндите!

Краток преглед на оператори за командата test

Оператор	Дава вредност „точно“ доколку ...	Операнди
-n	Операндот не е празна низа (нема должина нула)	1
-z	Операндот е празна низа (има должина нула)	1
-d	Постои именик чие име е <i>операндот</i> .	1
-f	Постои датотека чие име е <i>операндот</i> .	1
-eq	Операндите се цели броеви и се еднакви меѓу себе.	2
-ne	Спротивно од -eq	2
=	Операндите се еднакви (како низи од знаци)	2
!=	Спротивно од =	2
-lt	<i>operand1</i> е стриктно помал од <i>operand2</i> (двата операнди треба да се цели броеви)	2
-gt	<i>operand1</i> е стриктно поголем од <i>operand2</i> (двата операнди треба да се цели броеви)	2
-ge	<i>operand1</i> е поголем или еднаков на <i>operand2</i> (двата операнди треба да се цели броеви)	2
-le	<i>operand1</i> е помал или еднаков на <i>operand2</i> (двата операнди треба да се цели броеви)	2

Примери

- Примери
 - [-f "somefile"]
 - [-r "/bin/ls"]
 - [-w "/bin/ls"]
 - [-x "/bin/ls"]
 - [-n "\$var"]
 - ["\$a" = "\$b"]

Задача 1

- Да се напише командна процедура што ќе провери дали shell околината на најавениот корисник е BASH. Доколку не е, да ја испише работната shell околина преку променливата \$SHELL.

```
1 |#!/bin/bash
2
3 if [ "$SHELL" = "/bin/bash" ] #mora da ima prazni mesta
4     #kaj zagrada
5 then #mora da e vo nov red
6     echo "Vasata shell okolina e BASH (bourne again shell)"
7 else
8     echo "vasata shell okolina ne e BASH, tuku e $SHELL"
9 fi
10
```

Задача 2

- Да се напише командна процедура што ќе провери дали првиот аргумент од командна линија е 1, 2 или 3 и ќе испечати соодветни пораки. Доколку не е ниту една од овие вредности да испечати порака за грешка.

```
1  #!/bin/bash
2
3  if [ "$1" = "1" ]
4  then
5      echo "ja izbravte prvata opcija";
6  elif [ "$1" = "2" ]
7  then
8      echo "ja izbravte vtorata opcija";
9  elif [ "$1" = "3" ]
10 then
11     echo "ja izbravte tretata opcija";
12 else
13     echo "ne izbravte nisto"
14     echo "obidete se povtorno !!!"
15 fi
16
```

Оператори && и ||

```
[ -f "/etc/shadow" ] && echo "This  
computer uses shadow passwords"
```

```
#!/bin/sh  
mailfolder=/var/spool/mail/james  
[ -r "$mailfolder" ] || { echo "Can not read  
    $mailfolder" ; exit 1; }  
echo "$mailfolder has mail from:"  
grep "^From " $mailfolder
```

```
if [ -z "$DTHOME" ] && [ -d /usr/dt ] ;  
then DTHOME=/usr/dt ; fi
```

Оператор !

- **test ! expr** или **[! expr]**

```
if [ ! -d $HOME/bin ]  
then mkdir $HOME/bin  
fi
```

```
test ! -d $HOME/bin && mkdir $HOME/bin
```

Структурата case

- Структурата **case** овозможува споредување на дадена низа знаци со една од повеќе можни предложени низи. При споредувањето се користат и џокер знаците (како * или ?).
- Синтакса:

```
case niza in  
    shablon1) lista1 ;;  
    shablon2) lista2 ;;  
    ...  
esac
```

Пример за case

```
FRUIT=kiwi
case "$FRUIT" in
    apple) echo "Apple pie is quite tasty." ;;
    banana) echo "I like banana nut bread." ;;
    kiwi) echo "New Zealand is famous for kiwi." ;;
esac
```

```
if [ "$FRUIT" = apple ] ; then
    echo "Apple pie is quite tasty."
elif [ "$FRUIT" = banana ] ; then
    echo "I like banana nut bread."
elif [ "$FRUIT" = kiwi ] ; then
    echo "New Zealand is famous for kiwi."
fi
```

Задача 3

- Да се напише командна процедура што ќе провери дали првиот аргумент од командна линија е 1, 2 или 3 и ќе испечати соодветни пораки. Доколку не е ниту една од овие вредности да испечати порака за грешка. Задачата да се реши со case.

```
1  #!/bin/bash
2
3  case $1 in
4      1) echo "ja izbravte prvata moznost";;
5      2) echo "ja izbravte vtorata moznost";;
6      3) echo "ja izbravte tretata moznost";;
7      *) echo "ne izbravte nisto!!!"
8          echo "obidete se povtorno";;
9  esac
10
```

Задача 4

- Да се напише командна процедура smartzip што може да врши автоматско декомпресирање bzip2, gzip и zip компресирани датотеки.
- Пример за користење на file командата:
 - `$ file lf.gz`
 - `lf.gz: gzip compressed data, deflated, original filename, last modified: Mon Aug 27 23:09:18 2001, os: Unix`

Задача 4 – решение

```
1 |#!/bin/bash
2
3 filetype=$(file "$1")
4
5 case "$filetype" in
6 "$1: Zip archive"*) unzip "$1";;
7 "$1: gzip compressed"*) gunzip "$1";;
8 "$1: bzip2 compressed"*) bunzip2 "$1";;
9 *) echo "Datotekata $1 ne e vo potrebnit format"
10 exit 1;;
11 esac
12
```

Структури за повторување (while)

- Структурите за повторување, како во сите програмски јазици, овозможуваат повторување на цела низа изрази (команди) додека условот во структурата е точен.
- Синтакса:
while uslov
do lista
done
- Листата од команди се повторува се додека условот е точен, односно додека излезниот статус на условот е 0.

Вгнездени while јамки

```
while command1
  do
    list1
    while command2
      do
        list2
      done
    list3
  done
```

Задача 5

- Да се напише командна процедура што ќе одбројува одреден број секунди до почетокот на натпреварот.

```
1  #!/bin/bash
2
3  count=$1      #inicijalizacija na promenlivata count
4  # da bide ednakva na prviot argument
5
6  while [ $count -gt 0 ] #se dodeka count e pogolemo od 0
7  do
8      echo $count      #se pecati vrednosta na count
9      count=$(( $count - 1 )) #count se namaluva za eden
10     sleep 1          #se pauzira skriptata (spie) za 1 sekunda
11 done
12 echo "pocna natprevarot" #kraj
13
```

Задача 6

- Да се напише командна процедура што ќе ги прикаже на екран броевите од 0 до 20.

```
1 |#!/bin/bash
2
3 count=0
4
5 while [ $count -le 20 ]
6 do
7     echo $count
8     count=$(( $count + 1 ))
9 done
10
```

Структури за повторување (for)

- Синтакса:

```
for promenliva in zbor ...  
  do lista  
done
```

- Zbor е всушност цел израз или променлива што се заменува со нејзината вредност, по што се добива листа на вредности одделени со празни места. Структурата *for* ја доделува секоја од овие вредности на променливата *promenliva*, а потоа таа променлива се користи во телото на структурата. Делот *lista* (наредбите од телото на јамката) се извршува за секоја од вредностите доделени на *promenliva* поединечно.

```
for i in 0 1 2 4 3 5 8 7 9  
do  
  echo $i  
done
```

Наредба select

- Синтакса

select name in word1 word2 ... wordN

do list

done

- Извршување:
 1. Се прикажува секој елемент од листата со број пред него
 2. Се прикажува промптот
 3. Кога корисникот ќе внесе вредност таа се чува \$REPLY
 4. Откако ќе се изврши валидна селекција се извршуваат наредбите.
 5. Доколку нема наредба break, постапката продолжува од чекор 1.

Пример со select

```
select COMPONENT in comp1 comp2 comp3 all none
do
  case $COMPONENT in
    comp1|comp2|comp3) CompConf $COMPONENT ;;
    all) CompConf comp1
        CompConf comp2
        CompConf comp3
        ;;
    none) break ;;
    *) echo "ERROR: Invalid selection, $REPLY." ;;
  esac
done
```

- За да се смени промптот се користи:
 - \$ PS3="Napravete izbor => " ; export PS3

Контрола на јамки (break и continue)

```
while :  
do  
    read CMD  
    case $CMD in  
        [qQ] | [qQ][uU][iI][tT]) break ;;  
        *) process $CMD ;;  
    esac  
done
```

Контрола на јамки (break и continue)

```
for FILE in $FILES ;  
do  
if [ ! -f "$FILE" ] ; then  
    echo "ERROR: $FILE is not a file."  
    continue  
fi  
# process the file  
done
```

ПРАШАЊА?