

Командни процедури во UNIX (примери)

Оперативни системи
Аудиториска вежба 4



Задача 7

- Да се напише командна процедура што ќе ја прикаже содржината на повеќе текстуални датотеки.

```
#!/bin/sh
# Da se priazat soдрzinite na poveke tekstualni datoteki
# KORISTENJE: ./zad9.sh txt1.txt txt2.txt ...
# PRIMER: ./zad9.sh /home/ristes/*.txt

for datoteka in $*
do
    if [ -r "$datoteka" ]
    then
        echo "===== $datoteka ====="
        cat -n $datoteka
        #stava i broj pred redot
    else
        echo "ERROR: Ne se chita datotekata $datoteka"
    fi
done
```

Задача 8

- Да се напише командна процедура `rep` што ќе овозможи преименување на повеќе датотеки одеднаш со помош на наредбата `sed` за пребарување и замена на текст.

Задача 8 - решение

```
1 #!/bin/bash
2 # we have less than 3 arguments. Print the help text:
3 if [ $# -lt 3 ]
4 then
5   cat << HELP
6
7     zad10.sh -- renames a number of files using sed regular expressions
8
9     USAGE: zad10.sh 'regexp' 'replacement' files...
10
11     EXAMPLE: rename all *.HTM files in *.html:
12     ren 'HTM$' 'html' *.HTM
13
14   HELP
15   exit 0
16 fi
17
18
19 OLD="$1"
20 NEW="$2"
21
```

Задача 8 - решение

```
21
22 # komandata shift odstranjuva eden argument od listata
23 shift
24 shift
25
26 # $* sega gi sodrzhi site datoteki:
27 for f in $*
28 do
29     if [ -f "$f" ]
30     then
31         newfile=`echo "$f" | sed "s/${OLD}/${NEW}/g"`
32         if [ -f "$newfile" ]
33         then
34             echo "error"
35         else
36             echo "uspesno preimenuvanje na $f vo $newfile"
37             mv "$f" "$newfile"
38         fi
39     fi
40 done
41
```

Задача 9

- Напишете командна процедура која вредноста на првиот аргумент ќе ја зголеми 3.5 пати. Дозволете две места после децималната точка.

```
1  #!/bin/bash
2
3  PRICE=$1
4
5  #zad11
6  PRICE=`echo "scale=2; 3.5 * $PRICE"|bc`
7
8  echo $PRICE
9
```

Задача 10

- Напишете функција toLower која ги конвертира имињата на аргументите во мали букви и ги прикажува на екран.

```
1 |#!/bin/bash
2
3  #definiranje na funkcija
4  toLower() {
5      echo $@ | tr '[A-Z]' '[a-z]';
6  }
7
8  #povikuvanje na funkcija
9  toLower $@
10
11
```

Задача 11

- Напишете select јамка која ќе ја излиста секоја датотека во тековниот именик и ќе му овозможи на корисникот да ја погледне датотеката со одбирање на нејзиниот број.
- Користете го стрингот “Exit Program” за прекинување на јамката.
- Доколку корисникот одбере нешто што не е регуларна датотека, програмата треба да го идентификува тоа.

Задача 11 - решение

```
1  #!/bin/bash
2
3  allFiles=`ls`
4
5  select FILE in $allFiles "Exit Program"
6  do
7      if [ -z "$FILE" ]
8      then
9          continue
10     fi
11     if [ "$FILE" = "Exit Program" ]
12     then
13         break
14     fi
15
16     if [ ! -f "$FILE" ]
17     then
18         echo "$FILE is not a regular file."
19         continue
20     fi
21     echo $FILE
22     cat $FILE
23 done
```

Задача 12

- Напишете `sed` команда која како влез го има излезот од командата `ls -l`, но ќе ги печати само пермисиите и имињата на регуларните датотеки. Имениците, врските и специјалните датотеки не треба да се прикажуваат. Излезот од `ls -l` е следен:

```
-rw-r--r-- 1 ranga users 85 Nov 27 15:34 fruit_prices.txt
-rw-r--r-- 1 ranga users 80 Nov 27 13:53 fruit_prices.txt.7880
lrwxrwxrwx 1 ranga users 8 Nov 27 19:01 nash -> nash.txt
-rw-r--r-- 1 ranga users 62 Nov 27 16:06 nash.txt
lrwxrwxrwx 1 ranga users 8 Nov 27 19:01 urls -> urls.txt
-rw-r--r-- 1 ranga users 180 Nov 27 12:34 urls.txt
```

Додека, пак, вашиот излез треба да изгледа:

```
-rw-r--r-- fruit_prices.txt
-rw-r--r-- fruit_prices.txt.7880
-rw-r--r-- nash.txt
-rw-r--r-- urls.txt
```

```
ls -l | sed -e '/^[^-]/d' -e 's/ .*:[0-9][0-9] */ /'
```

Задача 13

- Напишете awk скрипта која ги печати полињата од датотеката fruit_prices.txt во обратен редослед.

- Датотеката fruit_prices.txt изгледа вака:

Fruit	Price/lbs	Quantity
Banana	\$0.89	100
Peach	\$0.79	65
Kiwi	\$1.50	22
Pineapple	\$1.29	35
Apple	\$0.99	78

- По извршување на скриптата излезот треба да биде:

```
Quantity Price/lbs Fruit
100 $0.89 Banana
65 $0.79 Peach
22 $1.50 Kiwi
35 $1.29 Pineapple
78 $0.99 Apple
```

Задача 13 - решение

```
1 |#!/bin/sh
2
3 |if [ $# -lt 1 ]
4 |then
5 |    echo "USAGE: `basename $0` files"
6 |    exit 1
7 |fi
8
9 |awk '{
10 |    for (i=NF;i>=1;i--) {
11 |        printf("%s ",$i) ;
12 |    }
13 |    printf("\n") ;
14 |}' $@
15
```

Справување со аргументи

```
USAGE="Usage: $0 [-c|-t] [file|directory]"
case "$1" in
    -t) TARGETS="-tvf $2" ;;
    -c) TARGETS="-cvf $2.tar $2" ;;
    *) echo "$USAGE"
        exit 0
        ;;
esac
```

```
USAGE="Usage: `basename $0` [-c|-t] [file|directory]"
```

Малку посложен пример

```
#!/bin/sh
USAGE="Usage: `basename $0` [-c|-t] [file|directory]"
if [ $# -lt 2 ] ; then
    echo "$USAGE"
    exit 1
fi
case "$1" in
    -t) TARGETS="-tvf $2" ;;
    -c) TARGETS="-cvf $2.tar $2" ;;
    *) echo "$USAGE"
        exit 0
        ;;
esac
tar $TARGETS
```

Функции

- Синтаксис

```
functionname( )
```

```
{
```

```
    # inside the body $1 is the first argument given to the
```

```
    # function, $2 is the second ...
```

```
    body
```

```
}
```

- Се повикува со

- `functionname`
- `functionname arg1 arg2 ... argN`

Корисни функции

```
lspath() {  
  OLDIFS="$IFS"  
  IFS=:  
  for DIR in $PATH ; do echo $DIR ; done  
  IFS="$OLDIFS"  
}
```

Се повикува со: lspath

Пример

```
#!/bin/sh
inc() {
    echo $(( $1 + $2 ))
}
if [ "$1" = "" ] || [ "$2" = "" ] || [ "$3" = "" ]
then
    echo USAGE:
    echo "counter StartValue IncrementValue EndValue"
else
    count=$1
    incValue=$2
    end=$3
    while [ $count -lt $end ]
    do
        echo $count
        count=$((inc $count $incValue))
    done
fi
```

The increment is defined first so we can use it

We echo the result of the first parameter plus the second parameter

We check to see that all the command line arguments are present

Rename are variables with clearer names

Loop while count is less than end

Call increment with count and value as parameters,

so that count is incremented by incValue

Опсег на променливи

```
#!/bin/sh
inc() {
    local value=4
    echo "value is $value within the function\\n"
    echo "\\$1 is $1 within the function"
}

value=5
echo value is $value before the function
echo "\\$1 is $1 before the function"
echo $(inc $value)
echo value is $value after the function
echo "\\$1 is $1 after the function"
```

Излез:

```
value is 5 before the function
$1 is 2 before the function
value is 4 within the function
$1 is 5 within the function
value is 5 after the function
$1 is 2 after the function
```

Задача 14

- Да се напише најкратка shell скрипта со која ќе се креираат два подименици (paren и nparen) во тековниот именик и истата ќе ги премести сите датотеки од тековниот именик во еден од двата подименици во зависност од бројот на карактери на датотеката.
- Притоа, датотеки кои имаат парен број на карактери треба да се пренесат во именикот paren, а оние со непарен број на бајти во именикот nparen.

Задача 14 - решение - прв начин

```
#!/bin/bash
mkdir .even .odd
for i in *
do
    case `wc -c <"$i"` in
        *[02468]) mv "$i" .even ;;
        *[13759]) mv "$i" .odd ;;
    esac
done
mv .even even
mv .odd odd
```

Задача 14 - решение - второй начин

```
#!/bin/bash
even=
odd=
for i in *
do
    case `wc -c <"$i"` in
        *[02468]) even="$even $i" ;;
        *[13759]) odd="$odd $i" ;;
    esac
done
mkdir even odd
mv $even even
mv $odd odd
```

Задача 15

- Напишете Shell скрипта со која во датотеката out.txt ќе го запише вкупното време, во минути, кое одреден корисник (прв аргумент на скриптата) бил најавен.
- Ако не е проследен аргументот, да се прикаже упатство за користење на скриптата.
- Доколку излезната датотека постои, потребно е да се пребрише.
- На крајот од скриптата да се прикаже содржината на датотеката out.txt.

Задача 15 - Анализа

121170	pts/25	92.53.4.153	Fri May 1 18:09 - 18:35	(00:26)
131513	pts/0	89.205.57.206	Fri May 1 18:08 - 20:29	(02:21)
131003	pts/0	79.141.126.75	Fri May 1 18:05 - 18:07	(00:01)
121079	pts/5	31.11.103.171	Fri May 1 18:02 - 21:01	(02:58)
125002	pts/24	78.157.14.93	Fri May 1 18:00 - 21:22	(03:21)
121021	pts/23	92.53.48.149	Fri May 1 17:58 - 18:29	(00:30)
141544	pts/22	85.30.78.174	Fri May 1 17:58 - 18:28	(00:30)
125018	pts/9	77.28.6.87	Fri May 1 17:55 - 21:20	(03:24)
131513	pts/20	89.205.57.206	Fri May 1 17:47 - 20:01	(02:14)
131004	pts/19	79.125.179.42	Fri May 1 17:47 - 18:37	(00:50)
141544	pts/17	85.30.78.174	Fri May 1 17:45 - 19:57	(02:12)
133011	pts/16	31.11.115.225	Fri May 1 17:39 - 18:34	(00:54)
131003	pts/5	79.141.126.75	Fri May 1 17:36 - 18:01	(00:24)
125015	pts/0	46.217.136.22	Fri May 1 17:33 - 18:03	(00:29)
133011	pts/16	31.11.115.225	Fri May 1 17:29 - 17:35	(00:05)

Задача 15 - Решение

```
#!/bin/bash
if [ $# -lt 1 ]
then
    echo "USAGE: `basename $0` username"
    exit 1
fi

logins=`last | grep $1`
times=`echo "$logins" | awk '{print $10}'`
timesCleared=`echo "$times" | sed -e 's/(//' -e 's/)//'`
minutes=`echo "$timesCleared" | awk -F: '{ print $1*60+$2}'`

total=0
for m in $minutes
do
    total=$(( $total + $m ))
done

echo $total > out.txt
cat out.txt
```


Задача 16

- Да се напише Shell скрипта која во датотеката out.txt ќе испечати колку деца процеси има секој од процесите на одреден корисник.
- Корисничкото име на корисникот се проследува како прв аргумент на скриптата.
- Ако не е проследен аргументот, да се прикаже упатство за користење на скриптата.
- Доколку излезната датотека постои, потребно е да се пребрише.
- На крајот од скриптата да се прикаже содржината на датотеката out.txt.

Задача 16 - Анализа

- Ако излезот на потребната варијанта на 'ps' е:

UID	PID	PPID	C	STIME	TTY	TIME	CMD
111xxx	9971	15761	0	16:42	pts/30	00:00:00	bash vtora.sh
111xxx	11434	15761	0	16:31	pts/30	00:00:00	bash vtora.sh
111xxx	12568	15761	0	16:34	pts/30	00:00:01	bash vtora.sh
111xxx	15760	15753	0	16:08	?	00:00:00	sshd: 111xxx@pts/30
111xxx	15761	15760	0	16:08	pts/30	00:00:00	-bash
111xxx	21199	9971	0	16:42	pts/30	00:00:00	[bash] <defunct>
111xxx	23329	15761	0	16:30	pts/30	00:00:00	bash vtora.sh
111xxx	26238	15761	0	16:43	pts/30	00:00:00	bash vtora.sh
111xxx	27977	15761	0	16:58	pts/30	00:00:00	bash vtora.sh
111xxx	30618	11434	0	16:31	pts/30	00:00:00	bash kol_1.sh
111xxx	30619	30618	0	16:31	pts/30	00:00:00	[bash] <defunct>
111xxx	30620	30618	0	16:31	pts/30	00:00:00	sed s/.*\./\./

- Датотеката out.txt треба да ја има следната содржина:

9971	1
11434	1
12568	0
15760	1
15761	6
21199	0
...	

Задача 16 - Решение

```
#!/bin/bash
if [ $# != 1 ]
then
    echo "USAGE: `basename $0` username"
    exit 1
fi

if [ -f out.txt ]
then
    rm out.txt
fi

for proc in `ps -ef | grep "$1" | awk '{ print $2; }'`
do
    count=0
    for pproc in `ps -ef | grep "$1" | awk '{ print $3; }'`
    do
        if [ $proc -eq $pproc ]
        then
            count=$(( $count + 1 ))
        fi
    done
    echo "$proc $count" >> out.txt
done
cat out.txt
```

Задача 17

- Да се напише Shell скрипта која ќе ги копира сите датотеки од именик зададен како прв аргумент кои започнуваат на број, по што следуваат само мали букви и имаат екстензија .out, во именик зададен како втор аргумент од командна линија.
- Потоа да се пресмета и да се испечати вкупната големина на ископираните датотеки за кои корисникот има привилегии за извршување.
- Доколку аргументите не се наведени, да се испише упатство за користење на скриптата, а ако не постои именикот зададен како втор аргумент, да се креира.

Задача 17 - Решение

```
#!/bin/bash
if [ $# -lt 2 ]
then
    echo "USAGE: `basename $0` sourcefolder/
                                destinationfolder/"
    exit 1
fi

from=$1
to=$2

if [ ! -d $to ]
then
    mkdir $to
fi
```

Задача 17 - Решение

```
files=`ls -l $from | grep '^-' | awk '{ print $9; }' |  
  grep '^[0-9][0-9]*[a-z]*\.out$`  
for file in $files  
do  
  cp ${from}${file} ${to}${file}  
done
```

```
filesX=`ls -l $to | grep '^-..x' | awk '{ print $5; }`  
total=0  
for i in $filesX  
do  
  total=`expr $total + $i`  
done  
  
echo $total
```

Задача 18

- Да се напише командна процедура користејќи ја командата за приказ на табела на процеси (top) ќе го прикаже вкупното време на извршување од почетокот до сега на командите стартувани од одреден корисник, кои се наоѓаат во првата серија (batch) на приказ на процесите.
- Името на корисникот се испраќа како аргумент на командната процедура.
- Доколку аргументите не се наведени, да се испише упатство за користење на скриптата.

Задача 18 - Анализа

Излез од командата: top -b -n 1

top - 20:45:52 up 17 min, 0 users, load average: 0.52,0.58,0.59

Tasks: 5 total, 1 running, 4 sleeping, 0 stopped, 0 zombie

%Cpu(s): 0.0 us, 0.8 sy, 0.0 ni, 99.2 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st

MiB Mem : 16151.3 total, 5489.2 free, 10438.1 used, 224.0 buff/cache

MiB Swap: 49152.0 total, 45148.1 free, 4003.9 used. 5582.6 avail Mem

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	10336	732	668	S	0.0	0.0	0:00.07	init(Ubuntu)
4	root	20	0	10320	576	440	S	0.0	0.0	0:00.00	init
7	root	20	0	10364	368	312	S	0.0	0.0	0:00.00	SessionLeader
8	asistent	20	0	18340	3952	3840	S	0.0	0.0	0:00.27	bash
131	asistent	20	0	18788	2008	1448	R	0.0	0.0	0:00.01	top

Задача 18 - Решение

```
#!/bin/bash

if [ $# -ne 1 ]
then
    echo "Usage: `basename $0` username"
    exit 1
fi

user=$1

total_time=`top -b -n 1 | grep $user | awk 'BEGIN {hours=0; minutes=0;
seconds=0;}          {split($11, h, ":"); hours += h[1]; split(h[2], ms, "."); minutes+=ms[1];
seconds+=ms[2];} END {print hours*3600+ minutes*60 + seconds;}`

echo $total_time
```

Задача 19

- Да се напише командна процедура која користејќи ја рутирачката табела од netstat командата ќе изброи колку од дестинациските адреси се мапирани на секој од физичките интерфејси.
- Резултатите да се запишат во излезна датотека дадена како аргумент на скриптата.
- Доколку излезната датотека постои, истата треба да се пребрише.
- Резултатите да се запишат во следниот формат:
Interface: <physical_interface> - Destinations: <count>

Задача 19 - Анализа

Излез од командата: `netstat -r`

Kernel IP routing table

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
192.168.20.0	0.0.0.0	255.255.255.0	U	0	0	0	eth2
192.168.20.1	0.0.0.0	255.255.255.255	U	0	0	0	eth2
192.168.20.255	0.0.0.0	255.255.255.255	U	0	0	0	eth2
224.0.0.0	0.0.0.0	240.0.0.0	U	0	0	0	eth2
255.255.255.255	0.0.0.0	255.255.255.255	U	0	0	0	eth2
192.168.19.0	0.0.0.0	255.255.255.0	U	0	0	0	eth3
192.168.19.1	0.0.0.0	255.255.255.255	U	0	0	0	eth3
192.168.19.255	0.0.0.0	255.255.255.255	U	0	0	0	eth3
224.0.0.0	0.0.0.0	240.0.0.0	U	0	0	0	eth3

Задача 19 - Решение

```
#!/bin/bash
```

```
if [ $# -ne 1 ];then echo "Usage $0 filename"; exit 1;fi
```

```
if [ -f $1 ];then rm $1;fi
```

```
netstat_output=$(netstat -r | grep ^[0-9]) # do not use the header
```

```
interfaces=$(echo "$netstat_output" | awk '{print $NF}' | sort | uniq)
```

```
for interface in $interfaces
```

```
do
```

```
    count=$(echo "$netstat_output" | grep -c " $interface$")
```

```
    echo "Interface: $interface - Destinations: $count" >> $1
```

```
done
```

Задача 20

- Да се напише командна процедура која ќе испечати информации за сите мрежни интерфејси на компјутерот. Потребно е да се прикаже IP адресата хардверската адреса на секој интерфејс.

- Резултатите да се запишат во следниот формат:

Interface: <interface_name>

<IP_address>

<hardware_address>

Задача 20 - Анализа

Излез од командата: ifconfig

eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 192.168.20.1 netmask 255.255.255.0 broadcast 192.168.20.255

inet6 fe80::4326:e4fd:4872:e702 prefixlen 64 scopeid 0xfd<compat,link,site,host>

ether 00:50:56:c0:00:01 (Ethernet)

RX packets 0 bytes 0 (0.0 B)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 0 bytes 0 (0.0 B)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

Задача 20 - Решение

```
#!/bin/bash
```

```
display_interface_info() {
```

```
    echo "Interface: $1"
```

```
    echo "-----"
```

```
    ifconfig $1 | grep -E "inet|HWaddr"
```

```
    echo "-----"
```

```
}
```

```
interfaces=$(ifconfig -a | grep -oE '^[a-zA-Z0-9]+' | grep -v 'lo') # skip the loopback address
```

```
for interface in $interfaces; do
```

```
    display_interface_info $interface
```

```
done
```

ПРАШАЊА

