2. Programare shell

Contents

2.1. Ce este un script (fişier de comenzi) shell

Un **script** (fişier de comenzi) este un fişier text care conţine în el:

- comenzi Unix;
- directive (ale interpretorului de comenzi shell) de control al fluxului execuţiei acestor comenzi.

Unscript se comporta, la randul lui, ca si o comanda shell. **Numele unui fişier script nu trebuie să respecte nici o cerinţă sintactică!**

Noi vom adăuga la numele de scripturi sufixul **.sh** ca o convenţie proprie de a ilustra conţinutul. In general, un script este folosit pentru a manevra fişiere din sistemul de fişiere. Dacă se doreşte intervenţia în interiorul fişierelor, de regulă a celor text, se folosesc comenzi filtru auxiliare: `grep, sed, cut, awk` etc.

Pentru testarea unor condiţii in structurile de control `if` sau `while` **se exploatează codul de retur al terminării comenzii precedente. Valoarea 0 a codului de retur reprezintă valoare de adevăr (TRUE), iar o valoare nenulă înseamnă FALSE**. De multe ori în condiţiile din if sau while se foloseşte, comanda `test expresie` (notat uneori mai elegant `[ expresie ]`) care întoarce codul de retur în funcţie de valoarea de adevăr a condiţiei din `test`.

Orice comanda shell se poate rula:

1. Direct la prompter în linia de comanda, aşa cum am vazut in seminarul precedent
2. Comanda se scrie intr-un script urmând a fi rulată odată cu execuţia scriptului, de care ne ocupăm în cele ce urmează.

Dacă `script` este numele unui fişier de comenzi din directorul curent, rularea acestuia se poate face:

1. `./script ...` sau `calebsoluta/script ...` dacă fişierul `script` are drepturi de execuţie. Pentru fixarea drepturilor, în particular şi a celor de execuţie, se foloseşte comanda `chmod`.
2. `script ...` daca `script` are drepturi de executie şi dacă directorul curent este inscris în `PATH`
3. `sh script ...` sau `sh caleabsoluta/script ...` indiferent dacă are sau nu drepturi de execuţie.

Prin `...` am notat: argumente ale comenzii, optiuni, fisiere, expresii, redirectari: < > >> <& >&

Daca o comanda trebuie reprezentata pe doua (sau mai multe) linii consecutive, toate liniile componente, cu exceptia ultimei, se incheie cu \<ENTER> (succesiunea de caractere '\\' '\n').

O succesiune de doua comenzi se separă prin: \n (linie noua) | (conectarea pipe a doua comenzi) & (executie in background), ; execuţie succesivă a două comenzi, && || a doua comandă se execută dacă prima s-a rterminat bine sau rău. Mai multe comenzi pot fi grupate prin (comenzi) sau { comenzi }

Intr-o linie, dacă apare caracterul # (diez), atunci tot restul liniei este interpretat ca şi **comentariu**. Este indicat (nu obligatoriu) ca scriptul shell sa inceapa cu un comentariu special

`#!/bin/sh sau #!/bin/bash sau #!/bin/ksh . . .`

Prin aceasta se indica sub controlul cărui Shell se va interpreta scriptul: `sh`, `bash`, `ksh` etc. In cele ce urmează vom folosi numai scriptul `sh`.

2.2. Structuri de control, comenzi specifice şi capcane în programarea shell

Structuri de control ale interpretorului sh (directivele) sunt: `if, case, for, while, until`. Pentru specificarea acestor structuri se folosesc cuvintele rezervate `if, then, elif, else, fi, case, in, esac, for, do, done, while, until`. Sintaxele acestor construcţii sunt:

if *lc* **then** *lc* [**elif** *lc* **then** *lc*]* [**else** *lc*]? **fi**
case *cuvant* **in** [*cuvant* [| *cuvant*]* **)** *lc* **;;**]+ **esac**
for *nume* **do** *lc* **done**
for *nume* **in** [*cuvant*]+ **do** *lc* **done**
while *lc* **do** *lc* **done**
until *lc* **do** *lc* **done**

Ce este scris intre [ ] este optional. Caracterele * ? + indica repetarea ca la expresiile regulare. Prin *lc* am notat, generic, orice succesiune de comenzi, legate între ele prin `; | || &&`

In strânsă legătură cu structurile de control acest context se pot folosi comenzile speciale `true, false, break, continue`. De asemenea, o serie de comenzi standard Unix sunt utile în principal în scripturi shell: `shift, read, readonly, sleep, exit, echo, test` (echivalent cu `"[ ..]"`), `export, expr, basename`, (apostroafele inverse) .

Semantica acestor construcţii este similară cu cea întâlnită în limbajele de programare. Exemplele care urmează vor ilustra funcţionarea lor.

!ATENTIE LA:

1. Echivalenţa **test conditie** şi **[condiţie]** şi a spaţierilor care îmbraca **[** şi **]**.
2. Terminatorul de comandă poate fi: \n ; & ;; || && deci grijă la mai multe comenzi pe aceeaşi linie.
3. Posibila apariţie în variabilele shell a valorilor string vid sau a valorilor ce conţin spaţii.
4. Specificul expresiilor regulare şi a celor extinse (-E) folosite de diverse comenzi (grep, sed, cut, sh, expr, test etc.).
5. Gruparea comenzilor prin ( ) sau { }.

Comanda **test** are un alias în **[** dacă este îmbrăcat în spaţiu şi dacă după condiţie apare **]**. Exemple:

```
florin@ubuntu:~$ test -f /etc/passwd && echo Sigur /etc/passwd este un fisier
Sigur /etc/passwd este un fisier
florin@ubuntu:~$ [ -f /etc/passwd ]&& echo Sigur /etc/passwd este un fisier
Sigur /etc/passwd este un fisier
florin@ubuntu:~$ [ -f /etc/passwd && echo Sigur /etc/passwd este un fisier
bash: [: missing `]'
florin@ubuntu:~$ if test -f /etc/passwd ;then echo Sigur /etc/passwd este un fisier; fi
Sigur /etc/passwd este un fisier
florin@ubuntu:~$ if [ -f /etc/passwd ];then echo Sigur /etc/passwd este un fisier; fi
Sigur /etc/passwd este un fisier
florin@ubuntu:~$ if [ -f /etc/passwd ;then echo Sigur /etc/passwd este un fisier; fi
bash: [: missing `]'
florin@ubuntu:~$ if [ -f /etc/passwd ];then echo Sigur /etc/passwd este un fisier fi
> fi
Sigur /etc/passwd este un fisier fi
```

Dacă există bănuiala că o variabilă stringul vid şi asta va putea afecta sintaxa, atunci se fac completări aşa încât să se evite apariţia stringului vid. De exemplu.

```
florin@ubuntu:~$ [ $A == "ceva" ] ; echo $?
bash: [: ==: unary operator expected
2
florin@ubuntu:~$ [ X$A == "Xceva" ] ; echo $?
1
florin@ubuntu:~$ A="Avion cu reactie"
florin@ubuntu:~$ echo $A
Avion cu reactie
florin@ubuntu:~$ echo $A | [ -f $A ] ; echo $?
bash: [: too many arguments
2
florin@ubuntu:~$ echo $A | [ -f "$A" ] ; echo $?
1
```

Gruparea comenzilor:

```
florin@ubuntu:/home$ pwd
/home/florin
florin@ubuntu:~$ pwd ; ( cd .. ; pwd ; ) ; pwd
/home/florin
/home
/home/florin
florin@ubuntu:~$ pwd ; { cd .. ; pwd ; } ; pwd
/home/florin
/home
/home
florin@ubuntu:/home$
```

Utilizarea parametrilor liniei de comanda `$0 $1 $2 . . . $9 $# $? $@ $*` se poate testa rulând scriptul `f`:

```
echo -n Daca directorul curent are continutul:; ls
echo si se da la linia de comanda: "sh f 1 2 \"3 4\" *.c *.py" avem:
echo arg0: $0
echo arg1: $1
echo arg2: $2
echo arg3: $3
echo arg4: $4
echo arg5: $5
echo arg6: $6
echo arg7: $7
```

```
echo arg8: $8
echo arg9: $9
echo Numarul argumentelor din linia de comanda \$\# : $#
echo Codul de retur al ultimului proces executat \$\? : $?
echo Iterare folosind \"\$\@\", vede cele 6 argumente:
for v in "$@" ; do echo $v ; done
echo Iterare folosind \"\$\*\", argumentele sunt concatenate intr-unul singur:
for v in "$*" ; do echo $v ; done
echo Iterare folosind \$\@ sau \$\* fara ghilimele, au acelasi comportament:
for v in $@ ; do echo $v ; done
echo Valorile \$\@ \$\* \"\$\@\" \"\$\*\", fara iterare, arata la fel: $@
```

Rularea are ca efect:

```
Daca directorul curent are continutul:a.sh AVION capitalize.py d f pa pi2.c pi.c r s x.sh
si se da la linia de comanda: sh f 1 2 "3 4" *.c *.py avem:
arg0: f
arg1: 1
arg2: 2
arg3: 3 4
arg4: pi2.c
arg5: pi.c
arg6: capitalize.py
arg7:
arg8:
arg9:
Numarul argumentelor din linia de comanda $# : 6
Codul de retur al ultimului proces executat $? : 0
Iterare folosind "$@", vede cele 6 argumente:
1
2
3 4
pi2.c
pi.c
capitalize.py
Iterare folosind "$*", argumentele sunt concatenate intr-unul singur:
1 2 3 4 pi2.c pi.c capitalize.py
Iterare folosind $@ sau $* fara ghilimele, au acelasi comportament:
1
2
3
4
pi2.c
pi.c
capitalize.py
Valorile $@ $* "$@" "$*", fara iterare, arata la fel: 1 2 3 4 pi2.c pi.c capitalize.py
```

2.3. Exemple de scripturi shell

2.3.1. Numărul total de linii de cod C din directorul dat ca parametru care nu sunt goale:

Vom da prima soluţie **grep**. Aici consideram liniile goale acelea care au doar un caracter: \n sau contin numai spatii si \t. Numărăm separat liniile nevide şi cele ce conţin numai spaţii şi \t.

```
#!/bin/sh
s=0
for f in $1/*.c ; do
    l=`grep -Ec "^.+$" <$f`       # Linii care au cel putin un caracter
    b=`grep -Ec "^[ \t]+$" <$f`   # Numai ' 'sau \t, macar un caracter
    s=`expr $s + $l - $b`
done
echo $s
```

Dacă se cere totalul acestor linii din toate fişierele C aflate în directorul dat şi din toţi subalternii lui, se va itera folosind în **for** comanda **find**.

```
#!/bin/bash
s=0
for f in `find $1 -type f -name "*.c"`; do
    l=`grep -Ec "^.+$" <$f`       # Linii care au cel putin un caracter
    b=`grep -Ec "^[ \t]+$" <$f`   # Numai ' 'sau \t, macar un caracter
    s=`expr $s + $l - $b`
done
echo $s
```

Soluţia a doua o dăm cu **awk**. In awk vom număra (în variabila t) numărul liniilor care au cel puţin un cuvânt (în sens awk). Totalul acestora se va da la ieşire la sfârşitul fiecărui fişier parcurs.

```
#!/bin/sh
s=0
for f in $1/*.c ; do
    l=`awk 'NF > 0 { t+=1}\
    END {print t}' <$f`
    s=`expr $s + $l`
done
echo $s
```

Soluţia a treia o dăm cu **sed**. Vom şterge din intrare spaţiile şi \t, rezultatul va intra în grep care va da la ieşire liniile nevide, iar wc va număra aceste linii.

```
#!/bin/sh
s=0
for f in $1/*.c ; do
        l=`sed 's/\s//g; s/\t//g' <$f | grep '^.' | wc -l`
        s=`expr $s + $l`
done
echo $s
```

2.3.2. Un exemplu de if: determinarea tipurilor argumentelor

Se parcurg argumentele liniei de comandă şi pentru fiecare din ele se specifică dacă este fişier, director, număr sau altceva:

```
#!/bin/bash
for A in $@; do
    if [ -f $A ]; then
        echo $A is a file
    elif [ -d $A ]; then
        echo $A is a dir
    elif echo $A | grep -Eq "^[0-9]+$"; then
        echo $A is a number
    else
        echo We do not know what $A is
    fi
done
```

2.3.3. Citiri de la consolă şi test de existenţă / citire a fişierelor

Se va solicita de la consolă nume, în mod repetat, până când se va da numele unui fişier care există.

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

```
F=""
while [ -z "$F" ] || ! [ -f "$F" ] || ! [ -r "$F" ]; do
    read -p "Da un nume de fisier:" F
done
```

sau

```
#!/bin/bash
F=""
while test -z "$F" || ! test -f "$F" || ! test -r "$F"; do
    read -p "Da un nume de fisier:" F
done
```

2.3.4. Monitorizarea stării unui director şi anunţul modificării lui

```
#!/bin/bash
D=$1
if [ -z "$D" ]; then
    echo "ERROR: No directory provided for monitoring" 1>&2
    exit 1
fi
if [ ! -d "$D" ]; then
    echo "ERROR: Directory $D does not exist" 1>&2
    exit 1
fi
STATE=""
while true; do
    S=""
    CONTENT=""
    for P in `find $D`; do
        if [ -f $P ]; then
            LS=`ls -l $P | sha1sum`
            CONTENT=`sha1sum $P`
        else
            LS=`ls -ld $P | sha1sum`
            CONTENT=`ls -l $P | sha1sum`
        fi
        S="$S\n$LS $CONTENT"
    done
    if [ -n "$STATE" ] && [ "$S" != "$STATE" ]; then
        echo "Directory state changed"
    fi
    STATE=$S
    sleep 1
done
```

Folosim `sha1sum` pentru a obține o sumă de control care este statistic imposibil să fie identică pentru conținuturi diferite. Verificăm detaliile fișierului (`ls -l`), precum și conținutul acestuia. Pentru directoare, folosim `-d` la `ls` pentru a lista detaliile directorului și nu conținutul acestuia și folosim ieșirea de `ls -l` pentru conținutul directorului. Controlul se repetă la fiecare secundă.

2.3.5. Anunţarea userilor cu directoare prea mari

```
#!/bin/bash
# check-home-dir-size.sh parent maxim
# parent este parintele directoarelor care trebuie verificate,
# iar directoarele au numele utilizatorilor.
# maxim este dimensiunea maxima in Ko
# Trebuie sa fie instalat mail, de exemplu sudo apt install mailutils
parent=$1
maxim=$2
cd $parent
```

```
for f in * ; do
    if [ ! -d $f ] ; then continue; fi
    lung=`du -c $f | awk '$2 == "total" { print $1 }'`
    if [ $lung -lt $maxim ] ; then continue; fi
    echo $f
    mail -s "Avertisment" $f@scs.ubbcluj.ro <<MESAJ
Directorul este prea mare
MESAJ
done
```

2.3.6. Anunţarea userilor cu prea multe procese

```
#!/bin/bash
# check-process-count.sh maxim
# maxim este numarul maxim de procese admis
maxim=$1
for user in `who | awk '{ print$1 }'` ; do
    if [ 0`ps -u $user | wc -l` -ge 0$maxim ] ; then # 0 daca cumva nu se da $1
    echo $user
    write $user <<MESAJ
Aveti prea multe procese
MESAJ
    fi
done
```

2.3.7. Distrugerea proceselor prea vechi

```
#!/bin/bash
# check-process-age.sh ore
# ore este numarul maxim de procese admis
# ps -eo pid,etime da pidul si durata de viata
for pid in `ps -eo pid,etime | awk 'length($2) == 8 && ore <= substr($2,1,2)+0 {print $1}' ore=$1` ; do
    echo $pid
    #kill -9 $pid
done
```

2.3.8. Anunţul încărcării exagerate a procesorului şi / sau a memoriei

```
#!/bin/bash
# check-server-load.sh addr procMax memMax
# addr adresa unde se trimite avertismentul
# procMax incarcarea maxima (procente) a serverului dupa 5 minute
# memMax incarcarea maxima (procente) a memoriei
procMax=$2
memMax=$3
while true ; do
    #sleep 300
    proc=`uptime | awk '{s=$(NF-1); s = substr(s,1,index(s,".")-1); print s}'`
    mem=`free | awk '$1=="Mem:" {print $2*100/$3}'`
    echo $proc
    echo $mem
    if [ $proc > $procMax -o $mem > $memMax ] ; then
    mail -s "Avertisment" $1 <<MESAJ
Server / memorie prea incarcat(a)
MESAJ
    fi
done
```

2.3.9. Verificare perechi fisier - lungime

Sa se scrie un fisier de comenzi care primeste ca parametri perechi formate dintr-un nume de fisier si un numar. Pentru fiecare astfel de pereche se va verifica daca dimensiunea fisierului coincide cu numarul respectiv si se va afisa un mesaj corespunzator.

```
#!/bin/sh
# Sa se scrie un fisier de comenzi care primeste ca parametri perechi formate
# din nume de fisier si un numar. Pentru fiecare astfel de pereche se va
# verifica daca dimensiunea fisierului coincide cu numarul respectiv, si se va
# afisa un mesaj corespunzator.
while true; do
   if [ -z $1 ] || [ -z $2 ]; then break; fi
   fisier=$1
   numar=$2
   if [ ! -f $fisier ]; then continue; fi # nu e fisier
   if [ `echo $numar | grep -Ec "[0-9]+"` -eq 0 ]; then continue; fi # NaN
   #lung=`ls -l $fisier | cut -d" " -f5`
   lung=`ls -l $fisier | awk '{print $5;}'`
   if [ `echo $lung | grep -Ec "[0-9]+"` -eq 0 ]; then continue; fi # NaN
   if [ $numar -eq $lung ]; then echo "$fisier are lungimea $numar"; fi
   shift 2
done
```

2.3.10. Numărul mediu de linii:

Sa se scrie un script shell care primeste ca parametru un nume de director si va determina numarul mediu de linii din toate fisierele text din acest director si din toate subdirectoarele acestuia.

```
#!/bin/sh
# Sa se scrie un script shell care primeste ca parametru un nume de director si
# va determina numarul mediu de linii din toate fisierele text din acest
# director si din toate subdirectoarele acestuia.
if [ $# -ne 1 ]; then echo "trebuie dat un director" 1>&2; exit 1; fi
if [ ! -d $1 ]; then echo "$1 nu exista sau nu este director" 1>&2; exit 2; fi
TotalLinii=0
TotalFisiere=0
find $1 -type f -print | while read Fisier; do
   if [ `file $Fisier | grep -ci "ASCII text"` -eq 0 ] ; then continue ; fi
   linii=`wc -l <$Fisier` # cu $Fisier (fara <), da la iesire si numele fisierului
   TotalLinii=`expr $linii + $TotalLinii`
   TotalFisiere=`expr $TotalFisiere + 1`
   echo $TotalFisiere >/tmp/${LOGNAME}TotalFisiere
   echo $TotalLinii >/tmp/${LOGNAME}TotalLinii
done
TotalFisiere=`cat /tmp/${LOGNAME}TotalFisiere`
TotalLinii=`cat /tmp/${LOGNAME}TotalLinii`
rm /tmp/${LOGNAME}TotalFisiere /tmp/${LOGNAME}TotalLinii
medie=`expr $TotalLinii / $TotalFisiere`
echo "Linii" $TotalLinii "Fisiere" $TotalFisiere "Medie" $medie
```

Variabilele `TotalLinii` şi `TotalFisiere` definite în liniile 7 şi 8 vor fi preluate în subprocesele din corpul while - do, dar din păcate valorile lor de acolo nu pot fi transmise spre procesul părinte. Din această cauză este nevoie de salvările din liniile 14 şi 15, respectiv obţinerea valorilor din liniile 17 şi 18. Directorul /tmp permite scrierea şi citirea de către orice user. De aceea, spre a evita suprapunerile, numele fişierelor de acolo se prefixează, de exemplu, cu numele de user.

2.3.11. Supravegherea continutului unui grup de directoare

Sa se scrie un script shell care monitorizeaza mai mute directoare. Monitorizarea se refera la aparitia unui fisier in aceste directoare. Numele directoarelor se dau la linia de comanda, iar numele fisierului urmarit se da de la tastatura. De asemenea, se da de la tastatura timpul, in secunde, intre doua cautari.

```
#!/bin/sh
# Sa se scrie un script shell care monitorizeaza mai mute directoare.
# Monitorizarea se refera la aparitia unui fisier in aceste directoare.
# Numele directoarelor se dau la linia de comanda, iar
# numele fisierului urmarit se da de la tastatura.
# De asemenea, se da de la tastatura timpul, in secunde, intre doua cautari.
echo -n "Introduceti numele fisierului: "; read nume t
echo -n "Introduceti timpul (sec) intre cautari: "; read timp t
while true; do
    find $* -type f -print | while read af; do
        nf=`echo $af | awk -F/ '$NF>1{print $NF;}'`
        if [ "$nf" = "$nume" ]; then echo $nume "in" $af; fi
    done
    sleep $timp
done
```

Remarcam utilizarea in find a variabilei **$*** , în loc de **$@**.

2.3.12. Selectare numai cuvinte din litere mici

Sa se scrie un script care primeste la linia de comanda doua nume de fisiere: `intrare iesire`. Scriptul preia `intrare` si pune in `iesire` liniile din `intrare` din care se retin numai cuvintele formate din litere mici. In `iesire` se vor ordona alfabetic liniile si se vor elimina dublurile.

Evident, pot fi mai multe soluţii. Noi vom da două soluţii, ambele folosind expresii regulare. O soluţie foloseşte `awk`, cealaltă `sed`.

```
#!/bin/sh
# Sa se scrie un script care primeste la linia de comanda doua nume
# de fisiere: intrare iesire
# Scriptul preia intrare si pune in iesire liniile din intrare
# din care se retin numai cuvintele formate din litere mici.
# iesire se vor ordona alfabetic liniile si se  vor elimina dublurile.

# varianta awk sort
awk <$1 '{for (i=1; i<=NF; i++)\
if ($i !~ /^[a-z]+$/) $i="";\
linie = ""; for (i=1; i<=NF; i++) if($i != "") linie = linie $i " ";\
if (linie != "") print substr(linie,1,length(linie)-1);}' |\
sort -u >$2.awk

# varianta sed sort
sed <$1 -e 's/\t/ /g'| # InlocuiesteTAB cu un spatiu \
sed -e 's/$/ /g'| # Adauga spatiu la sfarsit \
sed -e 's/^[^ ]*[^a-z ][^ ]* / /g'| # Primul cuvant din linie \
sed -e 's/ [^ ]*[^a-z ][^ ]*$/ /g'| # Ultimul cuvant din linie \
sed -e 's/[^ ]*[^a-z ][^ ]* / /g' | # Cuvintele terminate de spatiu \
sed -e 's/  / /g'| # Reduce numarul de spatii. De ce trebuie repetata? \
sed -e 's/  / /g'| # Reduce numarul de spatii \
sed -e 's/  / /g'| # Reduce numarul de spatii \
sed -e 's/  / /g'| # Reduce numarul de spatii \
sed -e 's/  / /g'| # Reduce numarul de spatii \
sed -e 's/^ //g' | # Sterge spatiul de la inceput \
sed -e 's/ $//g'|  # Sterge spatiul de la sfarsit \
sed -e '/^$/d' | # Sterge liniile goale \
sort -u >$2.sed
```

2.3.13. Primul cuvant de cel putin 5 caractere

Din directorul curent, să se determine primul fişier text care conţine o linie al cărei prim cuvant are cel puţin 5 caractere. Exemplul este dat pentru a testa `break 2` (ieşirea din două cilcuri interioare), care se pare că nu merge. Este suplinit prin break - uri succesive.

Selectarea primului cuvânt din linie se poate face în două moduri: folosind **`cut`** sau fara el.

```
#!/bin/sh
# Sa se caute in directorul curent primul fisier text care contine o linie
# in care primul cuvant este mai scurt de 5 caractere.

for x in * ; do
   if [ `file $x | grep -ci "ASCII text"` -eq 0 ] ; then continue ; fi
   #Variabila cuv1 retine primul cuvant de pe o linie, delimitator spatiu
   cat $x | while read cuv1 t ; do
   #cat $x | cut -d" " -f1 | while read cuv1 ; do
      #verificam daca linia nu e vida, respectiv lungimea primului cuvant
      if [ ! -z $cuv1 ] && [ `expr length $cuv1` -ge 5 ] ; then
         echo In $x s-a gasit $cuv1 cu lungimea `expr length $cuv1`
         #break 2 # Se iese din doua cicluri
         break  # Se iese din ciclul while
      fi
   done
   break # Se iese din ciclul for
done
```

2.3.14. Lista fişierelor cu anumite drepturi:

Sa se construiasca un fisier de comenzi care primeste ca parametru un nume de director (sa-l numim D) si un numar intreg (sa-l numim N). Pentru fiecare fisier din directorul D sau din subdirectoarele acestuia, pentru care userul are drepturi de citire si de executie, sa se afiseze (maximum) primele N linii.

```
#!/bin/sh
# Sa se construiasca un fisier de comenzi care primeste ca
# parametru un nume de director (sa-l numim D) si un numar intreg (sa-l numim N).
# Pentru fiecare fisier din directorul D sau din subdirectoarele acestuia,
# pentru care userul are drepturi de citire si de executie,
# sa se afiseze (maximum) primele N linii.

# verificam daca exista 2 parametrii in linia de comanda;
if [ ! $# -eq 2 ]; then echo "usage: shell1.sh director numar"; exit 1;fi
# verificam daca primul parametru este nume de director
if [ ! -d $1 ];then echo "$1 nu este director\!"; exit 1;fi
D=$1 # Numai pentru a fi in ton cu enuntul. Putem folosi si $1
N=$2 # Numai pentru a fi in ton cu enuntul. Putem folosi si $2

# find $D -perm -u=rx -type f
# va afisa toate fisierele din directorul $D si din subdirectoare
# pentru care user-ul (u) are drept de citire (r) si de executie (x).
# O alternativa pentru comanda find ar fi test cu optiunile -f,-r, -x

for fis in `find $D -perm -u=rx -type f`; do
        echo $fis           # afisam numele fisierului
        head -$N $fis       # afisam primele $N linii din fisier
done
```

2.3.15. Primele linii din fişiere cu cuvinte lungi:

Sa se creeze un fisier care contine numele tuturor fisierelor text, dintr-un director dat ca parametru si din subdirectoarele lui, care au cuvinte mai lungi de **n** caractere, unde **n** se citeste de la tastatura. Lista rezultata va fi ordonată alfabetic.

```
#!/bin/sh
# Sa se creeze un fisier care contine numele tuturor fisierelor text dintr-un
# director dat ca parametru si din subdirectoarele lui, care au cuvinte mai
# lungi de n caractere, unde n se citeste de la tastatura.
# Fisierul rezultat va fi ordonat alfabetic.

if [ $# -lt 1 ] ; then echo "introduceti cel putin un argument" ; exit 1; fi
if [ ! -d $1 ] ; then echo  "$1 nu e director" ; exit 2; fi
rm /tmp/${LOGNAME}numefisiere /tmp/${LOGNAME}rezultat >/dev/null 2>&1
echo -n "Introduceti n: "
read n
for fis in `find $1 -type f -print`; do
    if [ `file $fis | grep -c text` -ne 1 ] ; then continue; fi
    # variabilele n si fis din interiorul lui awk sunt altele decat cele din sh
    # La fel si $1 $2 . . .
    awk -v n=$n -v fis=$fis '\
{ for (i=1; i<=NF; i++) if (length($i) > n) print fis; }'\
<$fis >>/tmp/${LOGNAME}numefisiere # De ce am folosit /tmp ??
done
sort -u </tmp/${LOGNAME}numefisiere >/tmp/${LOGNAME}rezultat
cat /tmp/${LOGNAME}rezultat
```

2.3.16. Redenumirea fişierelor de un anumit tip si numararea aparitiilor unui cuvant

Sa se scrie un script shell care primeste 4 parametri: `director, extensie1, extensie2, cuvant`. Scriptul va redenumi toate fisierele cu extensia `extensie1` (dupa `.`) din director si subdirectoarele acestuia, dandu-le extensia `extensie2`. Va numara de cate ori apare `cuvant` in fiecare fisier.

```
#!/bin/sh
# Sa se scrie un script shell care primeste 4 parametri:
# director, extensie1,  extensie2, cuvant
# Scriptul va redenumi toate fisierele cu extensia extensie1 (dupa .)
# din director si subdirectoarele acestuia, dandu-le extensia extensie2.
# Va numara de cate ori apare cuvant in fiecare fisier.
if [ $# -ne 4 ]; then echo "director, extensie1,  extensie2, cuvant" >&2; exit 1; fi
if [ ! \( -d $1 \) ]; then echo "$1 nu exista sau nu este director" >&2; exit 2; fi
Total=0
find $1 -type f -print | sort | while read Fisier; do
    if [ `file $Fisier | grep -ci "ASCII text"` -eq 0 ]; then continue; fi
    ext=`echo $Fisier | awk -F. '$NF>1{print $NF;}'`
    if [ -z $ext ] || [ $ext != $2 ]; then continue; fi
    nume=`echo $Fisier | awk -F. '$NF>1{print substr($0,1,length($0)-length($NF));}'`
    apare=`grep -ci $4 $Fisier`
    Total=`expr $Total + $apare`
    echo $Total>/tmp/${LOGNAME}Total
    echo "In" $Fisier $4 "apare de" $apare "ori"
    echo $Fisier "se va redenumi in" $nume$3
    #mv $Fisier $nume$3 # Aici re face redenumirea
done
echo $4 "apare in total de" `cat /tmp/${LOGNAME}Total` "ori"
rm /tmp/${LOGNAME}Total
```

Remarcam separarea partilor din numele absolut al unui fisier folosind `awk`.

2.3.17. Aparitia unor cuvinte in linii din fisiere

Sa se scrie un script care primeste la linia de comanda triplete: `fisier, cuvant numar`
Pentru fiecare astfel de triplet, se vor afisa toate liniile din fisier care contin cuvant exact de numar ori.

```
#!/bin/sh
# Sa se scrie un script care primeste la linia de comanda triplete:
# fisier, cuvant numar
# Pentru fiecare astfel de triplet, se vor afisa toate liniile
# din fisier care contin cuvant exact de numar ori.
while true; do
   if [ -z $1 ] || [ -z $2 ] || [ -z $3 ]; then break; fi
   fisier=$1
   cuvant=$2
   numar=$3
   if [ ! -f $fisier ]; then continue; fi # nu e fisier
   if [ `file $fisier | grep -c "text"` -eq 0 ]; then continue; fi
   if [ `echo $numar | egrep -c "[0-9]+"` -eq 0 ]; then continue; fi # NaN
   awk '{k=0; for (i=1; i<=NF; i++) if ($i == cuv) k++;\
         if (k==num) print FILENAME ":" $0;}' num=$numar cuv=$cuvant $fisier
   shift 3
done
```

2.3.18. Reunirea fişierelor text

Se cere un script **`sh`** care primeste la linia de comandă un nume de director. Se cere ca toate fişierele cu conţinut text din acest director şi din descendenţii lui să fie concatenate într-unul singur, într-o formă tipăribilă. Fişierul în care se reunesc să înceapă cu un cuprins, în ordine alfabetică, care reperează fişierele componente făcând repertorizarea la nivel de linie sursă. Iată cum apare o porţiune din fişierul reuniune.

```
0 linii pana la ./b: ASCII text, with CRLF, LF line terminators
6 linii pana la ./c: ASCII text
9 linii pana la ./cl: POSIX shell script, ASCII text executable
38 linii pana la ./f: ASCII text
42 linii pana la ./f5: POSIX shell script, ASCII text executable
68 linii pana la ./lr: POSIX shell script, ASCII text executable
98 linii pana la ./pall: POSIX shell script, ASCII text executable
135 linii pana la ./pall.sh: POSIX shell script, ASCII text executable
Total general: 172 linii in 8 fisiere.


========================================
|| 0 linii pana la ./b: ASCII text, with CRLF, LF line terminators
========================================[
Sa se scrie un fisier de comenzi care preia un fisier de intrare dat ca
parametru si creeaza din el un alt fisier (al carui nume este dat ca
parametru) in care pastreaza doar cuvintele care contin litere mici. Fisierul
se va ordona alfabetic. Daca in rezultat exista linii consecutive identice, se
va pastra doar una dintre ele.

]========================================


========================================
|| 6 linii pana la ./c: ASCII text
========================================[
Sa se scrie un script shell care monitorizeaza aparitia in toate directoarele
date ca si parametru in linia de comanda a unui nume de fisier citit de la
```

```
tastatura.
 ]======================================


========================================
|| 9 linii pana la ./cl: POSIX shell script, ASCII text executable
========================================[

- - - - -

========================================
|| 135 linii pana la ./pall.sh: POSIX shell script, ASCII text executable
========================================[

- - - - -


]========================================
Total general: 172 linii in 8 fisiere.
========================================
```

Sursa scriptului este:

```
#!/bin/sh
# Sa se reuneasca intr-un fisier text, intr-o forma tiparibila,
# toate fisierele text din directorul $1.
# Fisierul text va incepe cu un cuprins al fisierelor continute.
if [ $# -ne 1 ]; then echo "folosire: pall director" >&2; exit 1; fi
if [ ! \( -d $1 \) ]; then echo "$1 nu exista sau nu este director" >&2; exit 2; fi
rm /tmp/${LOGNAME}Afisari /tmp/${LOGNAME}DeListat>/dev/null 2>&1
rm /tmp/${LOGNAME}/Total >/dev/null 2>&1
TotalLinii=0
find $1 -type f -print | sort | while read FISIER; do
    if [ `file $FISIER | grep -ci "ASCII text"` -eq 0 ] >/dev/null 2>&1
    #if file $FISIER | egrep "exec|data|empty|reloc|cannot open" >/dev/null 2>&1
      then continue
    fi
    NrLinii=`wc -l <"$FISIER"` # De ce este necesara delimitarea cu " ?
    Linie=${TotalLinii}" linii pana la "`file $FISIER`
    echo $Linie >/dev/tty
    echo $Linie >> /tmp/${LOGNAME}DeListat
    echo "\n\n========================================" >> /tmp/${LOGNAME}Afisari
    echo "|| "$Linie >> /tmp/${LOGNAME}Afisari
    echo "========================================[" >> /tmp/${LOGNAME}Afisari
    cat $FISIER >> /tmp/${LOGNAME}Afisari
    #pr -f $FISIER >> /tmp/${LOGNAME}Afisari
    echo "]========================================" >> /tmp/${LOGNAME}Afisari
    TotalLinii=`expr $TotalLinii + $NrLinii`
    echo $TotalLinii >/tmp/${LOGNAME}Total # De ce este necesar ?
done
TotalLinii=`cat </tmp/${LOGNAME}Total`
NrFisiere=`wc -l </tmp/${LOGNAME}DeListat`
Linie="Total general: $TotalLinii linii in $NrFisiere fisiere."
echo $Linie >/dev/tty
echo $Linie >>/tmp/${LOGNAME}DeListat
echo $Linie >>/tmp/${LOGNAME}Afisari
echo "========================================" >> /tmp/${LOGNAME}Afisari
cat /tmp/${LOGNAME}Afisari >>/tmp/${LOGNAME}DeListat
rm  /tmp/${LOGNAME}Afisari /tmp/${LOGNAME}Total
```

Pentru a nu perturba culegerea de fişiere, fişierele auxiliare necesare sunt memorate în directorul `/tmp`, prefixate cu numele userului (`/tmp` este directorul tuturor userilor). Mai întâi se verifică corectitudinea parametrului, apoi se sterg eventualele fişiere temporare vechi. Construcţia `>/dev/null 2>&1` ascunde fişierele de ieşire şi de erori standard. Lista tuturor fişierelor din director este dată de find, iar în ciclul while se lipesc fişierele de tip text. Decizia că un fişier conţine text este obţinută prin file şi grep.

Câteva variabile de mediu împreună cu efectul comenzii wc realizează calculele de numere de linii aferente. Derularea comenzii se vede pe tty, indiferent de ieşirea standard a scriptului. Fişierul `/tmp/${LOGNAME}DeListat` conţine reuniuneea fişierelor cu conţinut text din $1.

2.3.19. Analizaţi textul sursă al unui script

Considerând că directorul **DIR** conține o ierarhie de subdirectoare și fișiere text răspundeți la următoarele întrebări despre scriptul Shell UNIX de mai jos:

a) Ce va conține fișierul 1.txt după rularea scriptului?
b) Ce va conține fișierul 2.txt după rularea scriptului?
c) Explicați în detaliu expresiile regulare de pe liniile 2 și 5.

```
1 for f in `find DIR -type f`; do
2     if grep -q "^[^0-9]" $f; then
3         echo $f >> 1.txt
4     fi
5     if ! grep -q "[a-z]$" $f; then
6         echo $f > 2.txt
7     fi
8 done
```

Răspuns:

a) Căile (relative la DIR) ale tuturor fișierelor care conțin linii care nu încep cu cifră.
b) Calea (relativă la DIR) a ultimului fișier găsit de comanda find care nu conține linii terminându-se cu literă mică.

Linia 2: început de linie, interval de cifre negat. Linia 5: interval de litere mici, sfârșit de linie

2.4. Probleme propuse

1. Răspundeţi la următoarele întrebări, considerând o rulare a scriptului shell de mai jos:

a) De câte ori se afişează "OK"? Justificaţi răspunsul.
b) Care e valoarea variabilei f?
c) Care e valoarea variabilei d?
d) Care e valoarea variabilei x?
e) Care e valoarea variabilei y?

```
1 f=`find . -type f`
2 d=`find . -type d`
3 for x in $f; do
4     for y in $d; do
5         if [ $x = $y ]; then
6             echo "OK"
7         fi
8     done
9 done
```

2. Se d fişierul **abc.sh** conţinând scriptul Shell UNIX de mai jos. Răspundeţi la următoarele întrebări:
b) Ce se întâmplă dacă scriptul este rulat fără argumente?
c) Ce va tipări pe ecran rularea `./abc.sh f3` şi ce fişiere (nume şi conţinut) va crea, dacă f3 conţine "`abc 74 2-8 aa 3a =c b2`" şi rularea se face într-un director conţinând doar fişierele `abc.sh` şi `f3`?
d) Daţi un exemplu de fişier `f3` astfel încât rularea de la punctul precedent să creeze 4 fişiere noi astfel încât numele niciunuia să nu aibă prefixul `f3`.

```
1   n=0
2   for i in `cat $1`; do
3     c=`echo $i|cut -c1`
4     if echo $i|grep -q "^[0-9][0-9]*$"; then
5       echo $i >> $1.nr
6     elif echo $c|grep -q "[A-Za-z]"; then
7       echo $i >> $c
8     else
9       n=`expr $n + 1`
10    fi
11  done
12  echo $n
```

3. Sa se creeze doua fisiere, unul care va contine lista tuturor fisierelor dintr-un director si din subdirectoarele acestuia, iar al doilea lista tuturor subdirectoarelor. Fisierele vor fi ordonate dupa dimensiune, iar directoarele alfabetic.

4. Sa se calculeze suma fiecarei coloane de numere din oricate fisiere. Pentru fiecare fisier se vor afisa aceste sume, numarul maxim de coloane si numarul de linii.

5. Sa se scrie un fisier de comenzi care preia un fisier de intrare dat ca parametru si creeaza din el un alt fisier (al carui nume este dat ca parametru) in care pastreaza doar vocalele. Daca in rezultat exista linii consecutive identice, se va pastra doar una dintre ele.

6. Sa se scrie un fisier de comenzi care are ca parametri triplete formate dintr-un nume de fisier si doua cuvinte. Pentru fiecare astfel de triplet, se va inlocui in fisier ultima aparitie din fiecare linie a primului cuvant cu cel de-al doilea cuvant.

7. Sa se scrie un fisier de comenzi care preia un fisier de intrare dat ca parametru si creeaza din el un alt fisier (al carui nume este dat ca parametru) in care pastreaza doar cuvintele care contin litere mici. Fisierul se va ordona alfabetic, si se vor semnala liniile consecutive identice.

8. Sa se afiseze, pentru fiecare fisier din linia de comanda, cuvantul care apare de cele mai multe ori. Afisarea se va face in ordine descrescatoare a numarului de aparitii.

9. Sa se scrie un fisier de comenzi care va afisa toate fisierele dintr-un director si din subdirectoarele acestuia asupra carora au drepturi de scriere toate cele trei categorii de utilizatori. Aceste fisiere vor fi apoi redenumite, adaugandu-se sufixul .all.

10.Sa se afiseze, pentru fiecare fisier din linia de comanda, numarul liniei care apare de cele mai multe ori, afisarea facandu-se in ordinea decsrescatoare a numarului de aparitii.

11.Sa se scrie un fisier de comenzi care va afisa toate numele de fisiere dintr-un director dat ca parametru si din subdirectoarele sale, care au numele mai scurte de 8 caractere. Pentru aceastea, daca sunt fisiere text, li se vor afisa primele 10 linii.

12.Sa se afiseze, pentru fiecare fisier din linia de comanda, cuvantul care apare de cele mai multe ori intr-o aceeasi linie. Afisarea se va face in alfabetica dupa numele fisierului.

13.Sa se scrie un fisier de comenzi care creeaza un fisier care va contine toate fisierele dintr-un director dat ca parametru si din subdirectoarele sale pentru care membrii grupului nu au nici un fel de drept. Apoi, pentru aceste fisiere, se va da drept de scriere pentru membrii grupului.

14.Sa se scrie un fisier de comenzi care are ca parametri triplete formate dintr-un nume de fisier, un cuvant si un numar k. Pentru fiecare astfel de triplet, se vor afisa toate liniile fisierului care contin cuvantul respectiv exact de k ori.

15.Pentru toate fisierele text date in linia de comanda, sa se elimine toate liniile care contin un anumit cuvant apoi sa se afiseze liniile comune acestor fisiere.

16.Pentru fiecare parametru din linia de comanda, daca el reprezinta un fisier, se va afisa cuvantul de lungime maxima. Afisarea se va face in ordine decsrescatoare a lungimii cuvantului. Ceilalti parametri se vor scrie intr-un fisier cu numele 'eronat'.

17.Se cere un fisier de comenzi care afiseaza toate numele de fisiere dintr-un director dat ca parametru si din subdirectoarele acestuia al caror nume se termina in .me. Afisarea se va face in ordinea creerii lor. Pentru toate aceste fisiere, daca sunt fisiere executabile, se vor lansa in executie.

18.Sa se scrie un program de supraveghere care afiseaza pe ecran toti utilizatorii care lanseaza comanda 'who'.

19.Sa se creeze un fisier care contine numele tuturor fisierelor text dintr-un director dat ca parametru si din subdirectoarele acestuia care au cuvinte mai lungi de 15 caractere. Fisierul va fi ordonat alfabetic.

20.Sa se scrie un fisier de comenzi care are ca parametri cvadruple formate dintr-un nume de fisier, doua cuvinte si un numar k. Pentru fiecare astfel de cvadruplu, se va inlocui in fisier a k-a aparitie din fiecare linie a primului cuvant cu cel de-al doilea cuvant.

21.Se da un fisier care contine pe fiecare rand un nume urmat de 5 note. Se cere sa se construiasca un al doilea fisier care contine numele urmat de medie. Acest fisier va fi ordonat descrescator in functie de medie. Se va semnala daca un anumit nume apare pe mai multe linii in fisierul initial.

22.Sa se afiseze, dintr-o lista de fisiere, numele aceluia care contine numarul maxim de cuvinte distincte pe o linie. Totodata, se va crea un fisier care contine, pentru fiecare fisier, numarul liniei care are lungime maxima. Liniile fisierului vor fi ordonate descrescator dupa lungimea liniei.

23.Sa se scrie un program de supraveghere, care semnaleaza momentul in care se sterge un fisier dintr-un director dat ca parametru si din subdirectoarele acestuia. Se va afisa ora stergerii si dimensiunea fisierului.

24.Sa se afiseze, pentru fiecare fisier din linia de comanda, numarul de cuvinte care au lungimea mai mare decat un numar k, citit de la tastatura. Afisarea se va face decsrescator dupa numarul de cuvinte.

25.Sa se scrie un fisier de comenzi care primeste ca parametri fie nume de fisiere care exista, fie numere. Fiecare fisier se va afisa pe ecran (daca este fisier text, in caz contrar se va da un mesaj), iar pentru fiecare numar n se va crea un fisier 'f.n' care contine textul "Acest fisier nu a existat" urmat de patratul numarului respectiv.

26.Pentru fiecare fisier din linia de comanda se vor afisa toate liniile care sunt mai lungi de 10 caractere. De asemenea, se vor inlocui toate cifrele cu caracterul 'a'. Liniile unui fisier vor fi precedate de numele fisierului. Se va face o situatie finala cu numarul liniilor afisate din fiecare fisier, in ordine descrescatoare in functie de numarul liniilor.

27.Sa se scrie un fisier de comenzi care primeste ca parametri perechi formate din nume de fisier si un numar. Pentru fiecare astfel de pereche se va verifica daca dimensiunea fisierului coincide cu numarul respectiv, si se va afisa un mesaj corespunzator.

28.Sa se scrie un fisier de comenzi care primeste ca parametri perechi formate din nume de fisier si un numar. Pentru fiecare astfel de pereche se vor afisa toate liniile care au lungimea mai mare decat numarul respectiv. Liniile unui fisier vor fi precedate de numele fisierului. Se va face o situatie cu numarul liniilor afisate din fiecare fisier, ordonat descrescator in functie de acest numar.

29. Se da un fisier de intrare care contine cuvinte. Unele cuvinte sunt delimitate de carcterele "<" ">". Sa se afiseze fisierul, mai putin cuvintele marcate intre "<" si ">".

30. Se da un fisier care are pe fiecare linie mai multe cuvinte. Fiecare dintre aceste cuvinte contine 0 sau mai multe caractere "." Sa se elimine din cuvintele de rang par carcaterele ".". Cuvintele sunt despartite de spatii si pot contine orice carcter.

31. Sa se determine folosind comanda ping toate calculatoarele din reteaua scs care sunt pornite stiind ca aceste calculatoare au adresa ip de forma 192.168.144.x unde x = 1, 254.

32. Sa se determine din utilizatorii intrati in sistem in ultima luna, pe aceia care sunt in grupul ccubb (se poate folosi comanda last)

33. Sa se scrie un program shell care prelucreza un fisier si pastreaza in acest fisier doar cuvintele care contin doar litere mici.

34. La toate fisierele dintr-un director dat ca parametru sa li se adauge la sfarsit o linie care sa contina dimensiunea fisierului (impreuna cu dimensiunea informatiei care se adauga) si data ultimei modificari (ziua, ora si anul). Sa se testeze pentru directorul respectiv daca dupa o perioada de timp sau modificat fisiere.

35. Sa se sorteze toti utilizatori conectati in sistem descrescator dupa idle (timpul care au fost inactivi).

36. Sa se determine daca exista doi utilizatori conectati la sistem de la o aceeasi statie.

37. Folosind comanda last sa se determine numarul de statii distincte de la care sau realizat conectari la sistem, si numarul de conectarii de la fiecare statiie.

38. Sa se determine numarul de utilizatori distincti care lucreaza in sistem si numarul de sesiuni ale fiecarui utilizator.

39. Folosind comanda df sa se determine spatiul ocupat pe disc (pe o anumita partitie). Sa se determine numarul de fisiere de pe partitia corespunzatoare si dimensiunea medie a unui fisier.

40. Sa se faca topul primilor 20 de utilizatori din sistem din punct de vedere al dimensiunii cutiei postale. Acestora sa li se trimita un mail de atentionare.

41. Sa se determine utilizatorii din sistem care au drepturi pe directorul personal si pentru "others". Daca exista cel putin doua din cele trei drepturi setate sa li se trimita acestor utilizatori un mesaj de avertizare.

42. Sa se determine utilizatorii din sistem care au in directorul personal un director cu numele public_html. Sa se afiseze numarul total de utilizatori care au acest director si cat la suta din utilizatorii din sistem au acest director.

43. Sa se determine numarul de utilizatori distincti care au intrat in sistem in ultima luna, precum si numarul de conectari pentru fiecare. Sa se sorteze utilizatori dupa numarul de conectari.

44. Sa se determine numarul de utilizatori distincti care au intrat in sistem in ultima luna, precum si timpul total de conectare pentru fiecare. Sa se sorteze utilizatori dupa acest timp.

45. Sa se determine subdirectoarele dintr-un director dat ca parametru care contin un fisier care ocupa mai mult de jumatate din dimensiunea subdirectorului respectiv.

46. Sa se faca topul subdirectoarelor dintr-un director dat ca parametru dupa raportul "numar de fisiere continute" / "dimensiune subdirector".

47. Sa se scrie un program shell care pentru fiecare fisier cu drepturile 755 dintr-un director (si subdirectoarele sale) dat ca parametru ii schimba drepturile de acces in 744. Inainte de a schimba drepturile de acces, pogramul cere confirmare din partea utilizatorului (pentru fiecare fisier in parte).

48. Sa se scrie un script care primeste ca si parametru un nume de utilizator si cauta toate legaturile simbolice detinute de acel utilizator. Programul va numara aceste legaturi simbolice, afisand numarul lor pe ecran si din multimea aceasta de legaturi simbolice va redenumi in 'leg-simbolica' jumatate dintre ele (de pilda cele pare).

49. Sa se scrie un shell care primeste ca parametrii in linia de comanda un nume de director si oricate fisiere. Programul va verifica care dintre aceste fisiere sunt detip text si pentru acestea din urma va afla numarul de caractere pe care le contine. Daca numarul de caractere este par, programul va muta fisierul respectiv in directorul desemnat ca prim parametru.

50. Sa se scrie un program care va supraveghea toate conexiunile de la o masina (server) dat ca parametru in linia de comanda si va scrie intr-un fisier timpul (data) la care s-a realizat conexiunea. Daca in plus, portul la care se conecteaza masina pe serverul local este portul 23, programul va trimite un mail la o adresa (care vreti voi;) in care va scrie numele masinii (de la distanta) si timpul cand s-a realizat conectarea. (indicatie: se foloseste comanda `netstat`).

51. Sa se scrie un shell care primeste ca si parametru un nume de director. Programul va examina fiecare fisier din director (nu si subdirectoare) si va face urmatorul lucru: Fie fisierul 'exemplu.txt' care are N caractere. Programul va face in locul fisierului 'exemplu.txt' doua fisiere 'exemp' si 'lu.txt' (de fapt desparte numele original in doua), iar 'exemp' va contine primele N/2 caractere din 'exemplu.txt', iar 'lu.txt' va contine ultimene N/2 caractere din fisieul original 'exemplu'. Nota:se pot scrie mai multe fisiere, scenarii awk,..nu este obligatoriu sa se scrie un singur program shell.

54.Sa se scrie un program shell care numara toate fisierele cu drept de scriere pentru group si others dintr-un director care este dat in linia de comanda si subdirectoarele sale si numara si directoarele cu drept de executie pentru others din acest director si subdirectoarele sale.

55.Sa se scrie un script shell care tot citeste utilizatori din linia de comanda si pentru fiecare utilizator afiseaza numele real al acestuia, numarul de procese (daca are) si ce comenzi ruleaza.

56.Sa se scrie un program shell care primeste ca si parametru un nume de director si un fisier. Programul va afisa toate fisierele din director si subdirectoarele sale care au extensia .c si care sunt mai noi decat fisierul dat ca al doilea parametru.

57. Se cere un fisier de comenzi care are in linia de comanda un nume de fisier f si un numar n. Se cere crearea fisierului f prin concatenarea fisierelor f.1, f.2, ..., f.n. Daca unul din aceste fisere nu exista el va trebui creat si va trebui sa contina textul "Eroare: acest fisier nu a existat inainte de executia comenzii".

58. Pentru fiecare fisier ASCII dintr-un director dat ca parametru si din toate subdirectoarele lui, se vor afisa primele zece linii care contin un anumit text (dat ca parametru).