2.4. Shell – in UNIX

2.4.1.Tipuri de shell

- **bash – Bourne Again Shell -** conceput sub auspiciile GNU. Denumirea este un acronim de la *Bourne-Again Shell*, după numele lui Steve Bourne, autorul *shell*-ului *sh* pentru UNIX, predecesorul *bash*-ului; prompt: $, # ;
- **bsh - Bourne Shell** - in care este prezentat interpreterului de comenzi Shell; este varianta clasică, in care sunt scrise majoritatea fişierelor de comenzi; prompt: $, # ;
- **csh** - **CShel**l - are o sintaxă mai apropiată de limbajul C, oferind faţă se Bourne Shell (**sh**)
  - posibilitatea de a lucra cu variabile structurate, capacitatea de a memora şi reutiliza ultimele comenzi introduse,
  - posibilitatea de a defini şi utiliza pseudonime pentru facilitarea introducerii unor comenzi mai complexe.
    De exemplu comanda: **%alias lista ' ls -aFx** \!***'** introduce pseudonimul **lista** în locul comenzii **ls -aFx,** pseudonim ce poate fi utilizat ulterior ca o comandă.
  - posibilitatea trecerii unui proces din foreground la background, prin utilizarea comenzilor **fg** (foreground) respectiv **bg** (background); prompt: %, # ;
- **ksh- Korn Shell** -derivat din **sh**, a preluat o serie de caracteristici ale **csh**; facilităţi:
  - editarea liniilor de comandă similar cu editarea textelor la **vi** sau **emacs**;
  - utilizează pseudonimelor din **csh**, extinzând sfera de cuprindere şi la subprograme constituite din funcţii Shell;
  - grad înalt de portabilitate;
  - posibilitatea utilizării de variabile structurate asemănător **csh**; prompt: $, # ;
- **tcsh (tc shell); s.a.**

(*)

Documentare: man nume_shell

Intrare in shell: nume_shell

Iesire din shell: exit

	Bourne	C	TC	Korn	BASH
command history	No	Yes	Yes	Yes	Yes
command alias	No	Yes	Yes	Yes	Yes
shell scripts	Yes	Yes	Yes	Yes	Yes
filename completion	No	Yes*	Yes	Yes*	Yes
command line editing	No	No	Yes	Yes*	Yes
job control	No	Yes	Yes	Yes	Yes

2.4.2. Fisiere de configurare

.profile
/etc/profile
/etc/environment

Setările mediului:

env – afiseaza setarile mediului

stty - afiseaza

Viteza terminalului (se poate modifica)= bauds (B/s)

Caracterele de stergere erase si kill

Verificarea imparitatii, terminator de linie, ecoul caracterelor introduse

$env

(**env**ironment-echipament)

ce conține următoarele valori mai semnificative:

$HOME= referă catalogul alocat utilizatorului (*home directory*);

-**$SHELL**= interpreterul de comenzi utilizat;

$LOGNAME= numele utilizatorului în sistem;

$PATH= defineşte lista directorilor ce sunt parcurşi de Shell în căutarea fişierelor executabile corespunzător comenzii introduse;

$MAIL= numele fişierului ce conține mesajele primite de utilizator prin e-mail;

$TERM= tipul terminalului;

$PS1**= defineşte prompter1 asociat interpretorului (implicit este caracterul "**$");

$PS2= stabileşte al doilea prompter de continuare a unei comenzi (implicit este caracterul "**>**").

(*)

- Într-o linie de comandă construcțiile prefixate cu **$**, sunt înlocuite cu valorile variabilelor de mediu sau variabile Shell;
- interpreterul Shell conține şi anumite variabile predefinite, având următoarea semnificație:
  - **$0** este numele Shell-ului sau fişierului de comenzi indirecte, ca procese curente în curs de execuție;
  - **$1, $2, ... , $9** sunt parametrii transmişi procedurilor Shell pe linia de comandă;
  - **$#** numărul de parametri poziționali ai unei proceduri Shell;
  - **$?** codul de stare al ultimei comenzi executate;
  - **$$** identificator de proces al Shell;
  - **$!** identificatorul ultimului proces lansat în background;

$PS1="`pwd`>“ va seta prompter-ul la afişarea căii curente asemănător cu MS-DOS.

$PS1="$LOGNAME@`hostname`>“ va avea ca efect afişarea prompter-ului: userlogin@hostname>

carmen@infocib.ase.ro>

- *cuvintele precedate de $ sunt înlocuite cu valoarea variabilei respective (substituirea variabilelor) incluzându-le şi pe cele încadrate de ` `(substituirea comenzilor);*
- *rezultatul oricărei comenzi setează variabila de mediu $? prin: 0 - execuție normală; <>0 - eroare de execuție.*

(*)

- Execuția unei liste de comenzi ca un proces separat prin:

(listă_comenzi)

De exemplu: **$pwd;(cd /usr/user1;pwd);pwd**

prin care efectul comenzii **cd** este anulat de întâlnirea parantezei închise, deoarece cd a schimbat calea numai pentru subprocesul lansat pentru executarea acelei liste de comenzi.

- execuţia listei de comenzi în cadrul unui proces curent, atunci se va specifica acest lucru prin: {**listă_comenzi**}

Exemplu: **$pwd;{cd /usr/user1;pwd);pwd**

prin care efectul comenzii **cd** se păstrează şi după terminarea listei de comenzi cuprinse între acolade.

- funcţie Shell compusă dintr-o listă de comenzi inclusă între acolade

nume () {listă_comenzi}

apelul funcţiei se face prin **nume [parametru]**

când parametrii poziţionali $1, $2, ... , $9 sunt iniţializaţi cu valorile parametrilor actuali din linia de comandă.

2.4.3. Shell-ul - ca program

– După deschiderea sesiunii de lucru utilizatorul se găseşte sub controlul interpretorului de comenzi al Unix-ului, realizând următorii paşi:

- emite promt-ul,
- aşteaptă introducerea comenzii,
- decodifică linia de comandă (şiruri de caractere despărţite prin spaţii)- maxim 256 de caractere, iar restul se ignora;
- primul cuvânt din linia de comandă este interpretat ca nume_comandă ($nume_comandă [arg1] ...[arg n])
- se încearcă lansarea în execuţie a comenzii specificate (dacă este comandă internă o execută, dacă este externă o caută în PATH)
- aşteaptă sau nu terminarea execuţiei comenzii în funcţie de sintaxa liniei de comandă (cu sau fără & - în background)
- ciclul se reia până la CTRL+D, exit sau login

2.4.4. Shell-ul - ca proces

- – mecanismul intern de lansare în execuţie a comenzilor este:
  - După preluarea comenzilor de la utilizator şi identificarea ei , shell-ul creează un **fork** (adică se creează un proces nou identic cu cel iniţial, numit shell secundar)
  - Apoi se creează un **exec**, având ca parametrii numele comenzii şi argumentele acesteia
  - Astfel se creează 2 procese, în părinte se execută shell-ul iniţial, iar în fiu, programul asociat comenzii (figura...)
  - Părintele are 2 opţiuni:
    - aşteaptă terminarea fiului - prin directiva **wait**,
    - nu aşteaptă terminarea fiului – dacă comanda s-a lansat în background, cu **&**
- (vezi ../procese2/ex7.c, simularea comenzii ls –l nume_dir)

(*)

Simularea comenzii ls –l nume_dir

```
main()
{
  int p[2],i,j;
  pipe(p);
  if(fork()==0)
    {
      close(0);
      dup(p[0]);
      close(p[1]);
      execl("/bin/sort","/bin/sort",0);
    }
  else
   {
      close(1);
      dup(p[1]);
      close(p[0]);
      execl("/bin/ls","/bin/ls",0);
   }
}
```


2.4.5. Shell-ul - ca limbaj de comanda

- procesul de **preluare a linie de comandă** de la tastatură are 2 paşi:
  - *memorarea caracterelor în buffer* – se face până la <CR>; depăşirea buffer-ului nu este semnalată, având ca efect ignorarea comenzii;
  - *interpretarea comenzii* – după decodificarea comenzii, buffer-ul este eliberat, utilizatorul putând introduce altă comandă, fără a aştepta afişarea prompt-ului.
- ***Corecţia erorilor*** – ştergere caracter (BACKSPACE), ştergere linie (@); resetarea se face cu comanda stty

  $stty erase car_ştergere_caracter kill car_stergere_linie
- ***Expandarea numelor de fişier*** –

  * = înlocuieşte orice şir de caractere (rm *.c)

  ? = înlocuieşte orice caracter din poziţia respectivă (rm ?.c)

  [...]=descrie o listă de caractere ordonată lexicografic (litere sau cifre), sau o listă de caractere identificare individual, separate prin virgulă (rm ana[a-m][2,4,9])

  \ =determină impiedicarea shell=ului de a interpreta următorul caracter (rm pop*scu)

(*)

- *Redirectarea fişierelor standard* –
  - Fişiere standard – sunt deschide implicit de shell la deschiderea sesiunii de lucru şi atribuite terminalului; acestea sunt:
    - De intrare (0)
    - De ieşire (1)
    - De afişarea erorilor (2)
  - Operatorul „>”- redirectează fişierul standard de ieşire (exp: ls –l >fis – crează fis)
  - Operatorul „>>””- redirectează fişierul standard de ieşire (exp: ls –l >>fis – adaugă la fişierul existent fis)
  - Operatorul „<”- redirectează fişierul standard de intrare (exp: mail < fis – trimite e-mail lui user fişierul fis)
  - Operatorul „|” – conectarea ieşirii unei comenzi la intrarea alteia, fara crearea de fisiere intermediare;se numeste mecanismul de **pipe** (conductă); kernell-ul starteaza ambele procese in background, executandu-le sincronizat, asteptand iesirile pentru intrari.

(*)

- exp:

  $who |lpr

  trimite la imprimanta utilizatorii conectaţi în sistem; comanda who se execută concurent cu lpr, ieşirea lui who fiind conectată direct la intrarea programului lpr; comenzile sunt executate simultan, sincronizarea este realizată direct de kernel;

- exp:

  who > fis
  lpr < fis
  rm fis

  trimite la imprimanta utilizatorii conectaţi în sistem; comenzile se execută secvenţial, iar fişierul temporar fis este gestionat de utilizator;

(*)

– secvența de instrucțiuni care simulează pipe-ul shell (procese2/ex7.c):

```
main()
{
  int p[2],i,j;
  pipe(p);
  if(fork()==0)
    {
      close(0);
      dup(p[0]);
      close(p[1]);
      execl("/bin/who","/bin/who",0);
    }
  else
   {
      close(1);
      dup(p[1]);
      close(p[0]);
      execl("/bin/lpr","/bin/lpr",0);
   }
}
```


2.4.6. Shell-ul ca limbaj de programare

- Program = o secventa de comenzi shell (procedura shell);
- Este memorat in fisiere, la fel ca si programele scrise in diverse limbaje;
- Se executa la cerere;

Fisiere de comenzi indirecte = script-uri

- Una din cele mai importante functii ale shell-ului este executarea fisierelor de comenzi indirecte (script-uri)
- Un script =o procedura shell constituita ca un fisier text, care contine comenzi similare celor introduse interactiv
- Avantaj =>permite executia unui set complex de comenzi prin simpla introducere a numelui fisierului, care se comporta ca un fisier executabil, fara a avea cele 3 forme (sursa, obiect, executabil).

(*)

- **Variabile Shell**
  - Atribuire valori: var= valoare
  - Referirea variabilelor: $var
  - Pot fi exportate, cu `export var`
  - Sunt de 3 tipuri:
    - *Ale sistemului* – inițializate de shell la deschiderea unei sesiuni cu valori precizare în fișierele `/etc/environment, /etc/profile, .profile` din HomeDirectory; acestea sunt: `$HOME, $PATH, $PS1, $PS2, $LOGNAME, $MAIL, $SHELL, $TERM`

    - *Ale shell-ului* – modificate dinamic de interpretor, pe care utilizatorul nu le poate modifica; acestea sunt: $#, $?, $$, $!, $n (n=1-9), $@;

      Exp:

```
$cat variabile
echo Nr.argument= $#
echo Numele shell script-ului este $0
echo Argunemtele sunt: $@
echo PID-ul procesului este: $$
echo Argumentul al 2-lea este: $2
```

      Lansati-l cu nume procedura cu argumente, astfel: `$variabile cris -f 3 j`

(*)

- *Ale utilizatorului* – definite de utilizator; exp:

```
$TEST1="HELLO WORLD"
$echo $TEST1
HELLO WORD

$DEMO=HELLO
$echo $DEMO
HELLO
```

- Trimiterea var. comenzilor

```
$DEMO=/usr/bin
$echo $DEMO
/usr/bin
$cd $DEMO
$pwd
/usr/bin
```

(*)

- Trimiterea var. comenzilor ca parte a unei comenzi (cu{})

```
$name=gr1000
$echo $name
gr1000
$echo ${name}a
gr100a
$echo $name a
gr1000 a
```

- Trimiterea ieşirilor comenzilor către var (comanda tb.să fie între ` `, altfel o interpretează ca text (nu ' 'sau " ")

```
$TODAY=`date`
$echo $TODAY
Mon Mar 22 10:48:52 WET 2010

$TODAY='date'
$echo $TODAY
date

$DIR=`ls -la`
$echo $DIR
total 16 drwxr-xr-x 2 carmen
staff 256 Mar 22 10:24 .
drwxr-xr-x 3 carmen staff 256
Mar 22 10:15 .. -rw-r--r-- 1
carmen staff 52 Mar 22 10:26
interactiv -rw-r--r-- 1 carmen
staff 153 Mar 22 10:19
variabile
```


(*)

- Definirea var.de la tastatură

```
$cat interactiv
echo Introduceţi numele:
read nume
echo Hello $nume
```

- Operaţii aritmetice (nu se permit paranteze; se utilizează comanda `expr`; tb.să fie spaţiu între operator şi expresei; pt.înmulţire se foloseşte `\*`)

```
$num=2
$expr $num + 2
4

$expr $num +2
Syntax error

$num $num+2
2+2

$expr $num \* 2
4

$expr $num \*2
Syntax error

$rezultat = `expr $num \* 3`
$echo $rezultat
6
```

(*)

- Manipulări de date (concatenarea var.text cu alt text)

```
$nume=steve
$prenume=norton
$fullname=`echo Numele este $nume $prenume\`
$echo $fullname >temp
$cat temp
```

  - Caracterele \, ’, ” (\ -schimbă semnificația caracterului următor); exp:

```
$echo ”\” ”
”
$a=fred
$echo ’$a’
$a   (afişează textul $a)
$echo ”$a”
fred
```


(*) Comenzi (bsh)

- test,
- basename,
- dirname,
- eval,
- exec,
- exit,
- getopts,
- hash,
- kill,
- newgrp,
- set,
- shift,
- times,
- trap,
- type,
- ulimit,
- umask,
- wait

(*) Substitutii conditionate

$var	signifies the value of var or nothing, if var is undefined.
${var}	same as above except the braces enclose the name of the variable to be substituted.
${var-thing}	value of var if var is defined; otherwise thing. $var is not set to thing.
${var=thing}	value of var if var is defined; otherwise thing. If undefined $var is set to thing.
${var?message}	If defined, $var; otherwise print message and exit the shell. If the message is empty, print a standard message.
${var+thing}	thing if $var is defined, otherwise nothing.

(*) Structuri condiţionate Shell

if listă_comenzi; then listă_comenzi; else listă_comenzi; fi	select var in word1… word_n do listă_comenzi; com_1; continue; com_n; continue; done (apare PS2 >,PS3 #)
while listă_comenzi; do listă_comenzi; done	until listă_comenzi; do listă_comenzi; done
for var in word1… word_n do listă_comenzi; done	case word in pattern) listă_comenzi; esac

(*)

Exp:

```
$for number in 1 2 3
>do
> case $number in
> 1)echo one;;
> 2)echo two;;
> *)echo it is oen or two;;
> esac
>done
```

va afișa în shell-ul secundar, în funcție de selecție:

```
one
two
it is one or two
```


Lansarea in executie a script-urilor Shell

- **Lansarea în execuție a script-urilor Shell**
  - sunt 2 moduri de lansare în execuție:
    - $sh fis_script
    - $chmod a+x fis;
      $fis
  - parametrii de pe linia de comandă sunt referiți prin simbolurile $1... $9, $0 fiind numele comenzii (fișierului); dacă sunt mai mult de 9 parametrii, se folosește comanda internă `shift`, care copiază conținutul variabilei $n în $n-1

Exemple de script-uri

1- afişarea variabilei introduse de utilizator (nume)

```
echo Numele Dvs.?
read nume
echo Hello $nume
```

2- ştergerea fişierului de istoric al comenzilor (ex2)

```
(( nr_linii = 0 ))
if [ -s $HOME/.sh_history ];then
if (( (( nr_linii = nr_linii + `pg .sh_history | wc -l` ))
> $HISTSIZE));then
echo A T E N T I E !!!!!
echo -n "Fisierul de istoric al comenzilor contine:
$nr_linii de linii"
echo
echo  -n "Doriti sa-l stergeti acum (Y/N)?"
read acum

      if [ $acum != "Y" -a  $acum != "y" ] ;then  return 0;
      else rm .sh_history;
      fi
else return 0;fi
else return 0;fi
```


2.4.7. Transferul fişierelor pe/de pe server-ul de Unix

- se realizează cu FTP (File Transfer Protocol);
- este un model client –server;
  - *clientul* – pe staţia utilizatorului, poate fi:
    - comanda **ftp** - de al promt-ul DOS
    - un program sub Windows (exp: WinFTP)
    - browser-ului, folosind schema URL: ftp://user@infocib.ase.ro/cale_HOME/
  - *server-ul* FTP – rulează sub server-ul de Unix şi se numeşte ftpd (daemon ftp); permite conexiune pentru utilizator anonymous sau pe conturile existente

- există mai multe modalităţi de transfer; vom exemplifica modul din prompt MS-DOS
- din prompt-ul MS-DOS lansaţi comanda:

c:\>ftp

>?	**– vă arată toate subcomenzile ftp**
>o infocib.ase.ro	**-deschide conexiunea cu server-ul de ftp de pe infocib**
user: contul_vostru	
password: parola_voastra	
>lcd c:\dir_local	**– schimbă directorul de pe maşina locală, acolo unde se găsesc fişierele voastra/ sau unde doriţi să le puneţi pe cele aduse**
>bin	**– trecerea modului de transfer din ASCII în binar; se recomandă pt.fişiere ZIP, EXE, imagini etc.**
>hash	**– vizualizarea transferului fiecărui 2048 B**
>cd dir_server	**– schimbă directorul din home directory-ul user-ului**
>put fis.ext	**– pune fişierul din directorul local curent pe server-ul infocib, în directorul din home-ul utilizatorului**
>mput *.ext	**– multiple put – pune toate fişierele cu extensia .ext, cerând confirmare la fiecare**
>get fis.ext	**– ia de pe server şi pune pe local**
>mget *	**- multiple get**
>quit	– închidere sesiune