(bash) shell programozás
input feldolgozása

- 1. tokenekre bontás
- 2. tokenek parancsba szerkesztése
- Tokenek
  - szavak
    - » határolói: szeparátor vagy operátor
    - » szeparátor: space, tab, newline
  - operátorok
    - > >> >& >| < << <<- <& <>
    - | & ; ( ) || && ;; (( )) |&
- newline értelmezése
  - 1. ha a sor értelmezhető befejezett parancsként, akkor értelmeződik
  - 2. ha a sorban nincsen szó, akkor törlődik
  - 3. egyébként a newline érvénytelen, a parancs értelmezése folytatódik

Parancsok értelmezése

1. Kulcsszó
2. Nem idézett alias
3. Belső parancs
4. Függvény definíció
5. Beépített segédprogram
6. Egyéb beépített segédprogram
7. Abszolút pathnév
8. Futtatható program a path-ban

Kulcsszavak

case	function
do	if
done	in
elif	select
else	then
esac	until
fi	while
for	{}

nem POSIX: select

Belső parancsok

break	export
:	readonly
continue	return
. *file*	set
eval	shift
exec	trap
exit	unset

Beépített segédprogramok

alias	jobs
bg	kill
cd	newgrp
command	read
false	true
fc	umask
fg	unalias
getopts	wait

Összetett parancsok

- futtatás sub-shellben
  - (list)
- futtatás direct
  - { list ; }
  - { list & }
- feltételes végrehajtás
  - if
  - case
  - select (bash)
- ciklusok
  - for
  - while
  - until
- függvény definíció
  - function *name* { list; }
  - *name* () { list; }

Logikai értelmű vizsgálatok

- **test program**
  - test *args*
  - [*args*]
- **konstansok**
  - true, false
- **test program argumentumai**
  - string összehasonlítások
    - » s1 = s2
    - » s1 != s2
  - numerikus összehasonlítások
    - » n1 -eq n2 n1 = n2
    - » n1 -ne n2 n1 ≠ n2
    - » n1 -lt n2 n1 < n2
    - » n1 -le n2 n1 ≤ n2
    - » n1 -gt n2 n1 > n2
    - » n1 -ge n2 n1 ≥ n2

File vizsgálatok (test argumentumai)

- minden teszt *-operator file* alakú
- operátorok
  - s: a file létezik és nem 0 méretű
  - f: a file normál file
  - r: a file létezik és olvasható
  - w: a file létezik és írható
  - x: a file létezik és futtatható
  - d: a file létezik és directory
  - u: a file létezik és set-uid bit be van állítva
  - g: a file létezik és set-gid bit be van állítva
  - file1 -nt file2
    - » file1 újabb file2-nél
  - file1 -ot file2
    - » file1 régebbi file2-nél
  - file1 -ef file2
    - » file1 egy másik neve file2

test argumentumainak logikai kombinációja

- NOT `! expr`
- AND `expr1 -a expr2`
- OR `expr1 -o expr2`
- csoportosítás `( expr )`

Feltételes végrehajtás

```
if list
      then list
[elif list
      then list]
...
[else list]
fi
```

```
if test $1 = 'alma'
then echo '$1=alma'
fi
```

```
case word in
patterni)
      commi ;;
     ...
esac
```

```
case $fn in
*.c | *.cpp ) cppcompile $fn;;
*.pas       ) pascompile $fn;;
*           ) echo "Nem definialt $fn";;
esac
```


Ciklusok

```
for name [ in w1 .. wn ]
do
  commands
done
```

```
Pld.
for i in *
do
  echo $i
done
```

```
while list
do
  list
done
```

```
repeat list
do
  list
done
```


Parancsok futtatása

- direct futtatás (beépített parancsok)
- direct futtatás sub-shellben (új subshell indul)
  - ( )
  - &
  - `
  - |
- indirect futtatás sub-shellben (külső parancs, shell)
- Példa

```
cd ~/one
X=1; Y=2; Z=3
export X
(cd ../two; pwd; Y=7; echo $X $Y $Z)
pwd
echo $X $Y $Z
```

Kifejtés (expansion)

- Brace expansion
  - `a{d,c,b}e ----> 'ade ace abe'`
- Tilde expansion
  - login név
- Parameter expansion
  - `${parameter}`
- Command substitution
  - `$(command)`
  - `` `command` ``
- Arithmetic expansion
  - `$[expression]`
  - `$((expression))`
- Pathname expansion
  - `*`
  - `?`
  - `[abc] [a-z]`

Brace expansion

- nem POSIX és nem Bourne kompatibilis
- letiltható
  - bash -nobreaceexpansion
  - set +o breaceexpand
- szintaktika
  - {}zárójelpár
  - {}között legalább egy ,
- egymásba ágyazható
- végrehajtás balról jobbra
- eredménye több szó
- példa

  mkdir {Varga{Tibi,Kati,Judit,Kriszti},TothJutka}

  mkdir VargaTibi VargaKati VargaJudit VargaKriszti TothJutka

Tilde expansion

- HOME directory

  ~
- adott felhasználó HOME directory-ja

  ~*user*
- HOME directory alatti directory-k

  ~/alma/korte

  ~lakatos/alma/korte
- aktuális directory (PWD változó értéke)

  ~+
- előző directory (OLDPWD változó értéke)

  ~-
- ha *user* nem létezik nincs ~ helyettesítés

Parameter expansion 1.

- alap (basic expansion)

  $param

  ${param}
- feltételes (conditional expansion)

  ${param:-word}
  - ha param létezik és értéke nem zérus - $param
  - ha param nem létezik - word

  ${param:?word}
  - ha param létezik és értéke nem zérus - $param
  - ha param nem létezik vagy zérus akkor
    - » hibaüzenet, ami word
    - » ha nem interaktív shell, akkor kilép

  ${param:+word}
  - ha param létezik és értéke nem zérus - semmi
  - ha param nem létezik - word

Parameter expansion 2.

- opcionális értékadás (csak változókra)

  ${param:=word}
  - ha param létezik és értéke nem zérus - $param
  - ha param nem létezik - word; mellékhatás: param értéke word
- hossz meghatározás

  ${#param}
  - param hossza (karakterek száma)
- mintaillesztés (pat-re pathname expansion)

  ${param#pat}
  - param eleje illeszkedik - (legrövidebb)

  ${param##pat}
  - param eleje illeszkedik - (leghosszabb)

  ${param%pat}
  - param vége illeszkedik - (legrövidebb)

  ${param%%pat}
  - param vége illeszkedik - (leghosszabb)

Command substitution parancshelyettesítés

- a parancs output helyettesítődik
- néhány karakter megtartja speciális jelentését
  - $ ` \
- kétféle szintaktika
- `command`
  - a régebbi Bourne shell-ben csak ez működik
- $(command)
  - egymásba ágyazható
- példák

  echo `echo `echo korte` alma`

  bash: alma: command not found

  echo korte

  echo $(echo $(echo korte) alma)

  korte alma

Arithmetic expansion

- aritmetikai értelmezés
  $[expression] (csak bash)
  $((expression))
- egymásba ágyazható
- oktális számok 0 prefix
- hexadecimális számok 0x vagy 0X prefix
- tetszőleges számrendszerű számok
  [base#]number
- operátorok, mint C
- példa
  a=b+9; b=-2; echo $((a*5))
  35

Aritmetikai operátorok

- +	unary minus and plus
! ~	logical and bitwise negation
* / %	multiplication, division, remainder
+ -	addition, subtraction
<< >>	left and right bitwise shifts
<= >= < >	comparison
== !=	equality and inequality
&	bitwise AND
^	bitwise exclusive OR
|	bitwise OR
&&	logical AND
||	logical OR
= *= /= %=	assignment
+= -= <<= >>=	
&= ^= |=	

Néhány előre definiált változó

- POSIX
  PATH, CDPATH, HOME, ENV, PS1, PS2, IFS, ...
- directoryk
  PWD, OLDPWD
- prompt
  PROMPT_COMMAND
- rejtett file-ok file-név helyettesítésnél
  glob_dot_filenames
- változók értékadása
  PROMPT_COMMAND=alma
- változók beállítása
  glob_dot_filenames=
- változók törlése
  unset glob_dot_filenames

Shell belső parancsok 1.

- üres utasítás (az argumentumok kiértékelődnek!)
  : [args]
- shell script behelyezése (include)
  . *file* [args]
  source *file* [args]
- parancs futtatása subshell nélkül
  exec *command*
- parancs képzés, majd végrehajtás
  eval [args]
  - » alma=korte
  - » eval $alma
  - » bash: korte: command not found
- kilépés a shellből
  exit [n]

Shell belső parancsok 2.

- ciklusból kilépés (több szintet is)
  break [n]
- ciklus következő (akár több) iterációja
  continue [n]
- függvény viszatérési értéke
  return [n]
- változók exportja (környezeti változók)
  export [name=[word]] ...
- változók attribútumai (nem POSIX)
  declare [-frxi] [name=value]
  typeset [-frxi] [name=value]
  readonly [-f] [name...]
- változók törlése
  unset [name...]

Shell belső parancsok 3.

- új változók automatikus/nem automatikus exportja
  set ±a
- pozicionális paraméterek megváltoztatása
  set arg1 arg2 ...
- pozicionális paraméterek léptetése
  shift [n]
- terminálódott háttérprocesszek jelzése azonnal/következő promptnál
  set ±b
- file-név helyettesítés letiltása/engedélyezése
  set ±f
- command line editor beállítása
  set -o vi
- szignálok kezelése
  trap [command] [sigspec]

Shell beépített parancsok 1.

- rövidítések (alias) megadása, kiírása
  alias [name=value]
- alias megszüntetése
  unalias [name]
- jobok kiiratása
  jobs
- job háttérbe helyezése
  bg [jobspec]
- job előtérbe helyezése
  fg [jobspec]
- processz befejezésének megvárása
  wait [pid]
- szignál küldése egy job-nak
  kill [-signspec] [pid|jobspec]
  kill -HUP 14745

Shell beépített parancsok 2.

- parancs futtatása
  command command
- directory váltás
  cd [dir]
- aktuális directory kiiratása
  pwd
- változó értékének interaktív olvasása
  read [name...]
- új file létehozásának maszkja
  umask [mode]
- kiírás a standard outputra
  echo [-neE] [args...]
- kilépés a login shellből
  logout

Shell beépített parancsok 3.

- directory stack kiiratása
  dirs
- directory stack növelése
  pushd [dir]
- directory stack csökkentése
  popd [+/-n]
- directory stack rotálása
  pushd +/-n
- aritmetikai kifejezés kiértékelése
  let expr ...
- rövid információ a beépített(!) parancsokról
  help pattern
- command history kiiratása
  history

Példák 1.

- jelezzük, ha péntek 13. van

1. változat

```
set `date`
if test $1 = 'Fri'
then
if test $3 = '13'
then echo Vigyazz!
fi
fi
```

2. változat

```
alma=`date '+%a%d'`
if test $alma = 'Fri13'
then echo Vigyazz
fi
```


Példák 2.

file-ok átnevezése .htm-ről .html-re

```
for fn in *.htm
do
mv $fn $(basename $fn .htm).html
done
```

file-ok összecsomagolása

```
echo '# To unbundle, sh this file'
for i
do
        echo "echo $i 1>&2"
        echo "cat >$i <<'End of $i'"
        cat $i
        echo "End of $i"
done
```


awk

- sorokat olvas
- alapvető feladat: mintaillesztés
- minták: reguláris kifejezés
- illeszkedő mintákra meghatározott tevékenység
- input egységes: sorok
- sorokon belül paraméterek
- awk-n belül soronként $1, $2, ...
- definiálható szeparátor
- tevékenység C szintaktikával
- változók definiálása nem szükséges
- asszociatív tömbök

awk - szintaktika

- lehetséges sorok

```
BEGIN              { action }
                   { action }
pattern
pattern            { action }
pattern1, pattern2 { action }
END                { action }
```

- előre definiált változók

```
FS    szeparátor
NF    mezõk száma a rekodban (sorban)
NR    az olvasott rekordok (sorok) száma
ARGC  argumentumok száma
ARGV  argumentumok tömbje
```


awk - példa

- írjuk ki az azonos login nevű felhasználókat,
- majd írjuk ki a login neveket a hozzá tartozó UID-al

```
BEGIN {
      FS = ":"
}
{ if (user[$1] {
      printf "%s duplicated\n", $1
      }
      user[$1] = $3
}
END {
      for (i in user)
            printf "%s = %s\n", i, user[i]
}
```