6. óra

## Ami eddig volt

Számítógép architektúra, alkotóelemek, hardverek, szoftver (gépi kód, OS, unix alapok, grep-szűrő és kapcsolói)

### Regexp ("reguláris kifejezés") to validate input

Általában nem fogadunk el tetszőleges bemenetet. Valamilyen feltételnek meg kell felelni. (e-mail, bankszámla, telefon, tizedes jegy, stb.)

# Shell script programozása

## Változók

Hasonló a környezeti változókhoz. Szövegesek. (Csak karakter alapúak.) Látja magán kívül a környezetit. Se szöveges, se aritmetikai műveleteket nem támogat! Amit egymás után írunk, az egybe tartozik.

### Értékadás, összefűzés

x=alma; y=$x+fa; echo $y; #eredménye: "alma+fa", nem "almafa". Helyesen: y=${alma}fa;

a=5; b=$a+1; echo $b; #eredménye: "5+1", nem "6", nem is "51".

### Hogyan kell akkor összeadni?

BASH: let utasítás. a=2; let b=a+1; b ekkor 3 lesz. De ezt csak a BASH tudja.

BASH előtt: expr utasítás. expr $a op $b

op(erátor) lehet: (<, <=, >, >=, !=, =,) +, -, \*, /, % (mod). Pl.: expr 3 \\* 4 ("\*" wildcard, "/" comment, "<" input, and ">" output have to be escaped!)

Logikai operátorokat inkább más, erősebb, bővebb utasítással fogjuk kiértékelni: test.

### És a gyökvonás, a cosinus?

bc parancs. (bináris kalkulátor) szűrő-szerű program. C típusú szöveget vár bemenetként.

echo 2\*16 | bc #eredménye:32

Van benne: ciklus, szögfüggvény, önálló függvénydefiníció, stb. Példa:

#!/usr/bin/bc -l
define fakt(x){
if x<=1 return 1;
return fakt(x-1)\*x;
}
#s=sin c=cos t=tan
#további parancsok: man bc

## Paraméterek

parancs paraméter1 paraméter2 … paraméter(n)

$0=parancs $1=paraméter1 $2=paraméter2 ${n}=paraméter(n)
$#=paraméterek száma (n)
10-nél több paraméter esetén: $10 nem jó, mert $1-hez fűzné a 0-t. Helyette ${10}. Vagy "shift" utasítással kidobjuk a $1-et és helyébe rakjuk a $2-t. $3 helyébe $2, stb.
$\*=összes paraméter egyben: paraméter1 paraméter 2 … paraméter(n)
"$@" külön a paraméterek, ""-es bemenet esetén. (???)
A $-os változók környezeti változók.

for ciklus lefutása fű fa 'szál fa' bemenet esetén:
$\* 4
"$" 1
"$@" 3

## Eredmény

Eredmény a $? környezeti változóban található. Minden utasításnak, parancsnak van eredménye. Ha nincs, akkor is van. A script utolsó utasításának eredménye lesz az.
exit 0; # nincs hiba.
Ha az eredmény 0, logikai igaz. Minden más érték (1-255) hamis. Ez pont a C stílus ellentétje. (Ott "hiba volt-e". Itt "sikeres-e".)

## Input-Output

echo kiír, read beolvas, line sort beolvas.

read alma #alma változóba olvas.

A line szűrőként is használható, a read nem.

Parancsbehelyettesítéses for ciklus-sal kombinálva használjuk.

név=`echo $alma | cut -f1 -d\& | cut -f2 -d=`

$alma "&" határoló (delimiter) karakter mentén felszeletelt első mezejét még "=" jelek mentén is szeleteljük, ebből a 2. mezőt vesszük le. Ez lesz a név értéke parancsbehelyettesítés (`) miatt.
Ez HTML CGI paraméterekből kivesz login nevet (és jelszót).

## Elágazások

test utasítás vagy annak sajátos rövidítése: [ … ] logikai utasítás. Fontos a helyköz [ után és ] előtt! Mert paraméterelválasztó szereppel bírnak!
0 az igaz 1 a hamis.
if
 [ … ]
then
 echo Kisebb.
else
 echo Nagyobb.
fi #if lezárása a fi.

### Numerikus vizsgálat

-lt, -gt, -le, -ge, -eq, -ne
if [ $x -lt 5 ]

### String vizsgálat

=, !=

-z string igaz, ha üres (hossza 0).

-n string igaz, ha nem üres (hossza nem 0).

### Fájl, mappa vizsgálat

-f fájl létezik-e
-d könyvtár létezik-e

### Több logikai művelet összekapcsolása

-o vagy
-a és
! tagadás

## Ciklusok

for i in $\*
do
… #pl echo $i
done

## Könyvtári környezet megismerése

Minél több unix shell parancs megismerése.

## Feltételes parancsvégrehajtás

if sikeres then másikparancs fi
sikeres && másikparancs

if nemsikeres then másikparancs fi
nemsikeres||másikparancs

## Többirányú elágazás (case)

(C-ben switch.)

case $alma in
idared) echo az alma idared
;;
golden) echo az alma golden
;;
\*)
;;
esac

## Példák

if
 who |grep jani /dev/null
then
 echo Jani be van jelentkezve.
else
 echo Jani nincs bejelentkezve.
fi

read x
case $x in
[dD]\*) date;;
[wW]\*) who;;
…

for i in alma körte barack
 do
 echo $i
 done

for i in $seq(10)
 do
 echo $i
 done

for i in `who`
 do
 echo $i
 done

while
 echo -n Írd be a neved:
 read nev
do
 echo nev
done

until
 [ $v = "nem" ] #hamisnak kell lenni
do
 … Folytassuk? \(igen/nem\) … #fontos a \( és \), mert csoportosítás lenne különben.
done
Shell paraméterek

-v kiírja a parancsot végrehajtás előtt.
-x TODO
-n szintaktikai ellenőrzés