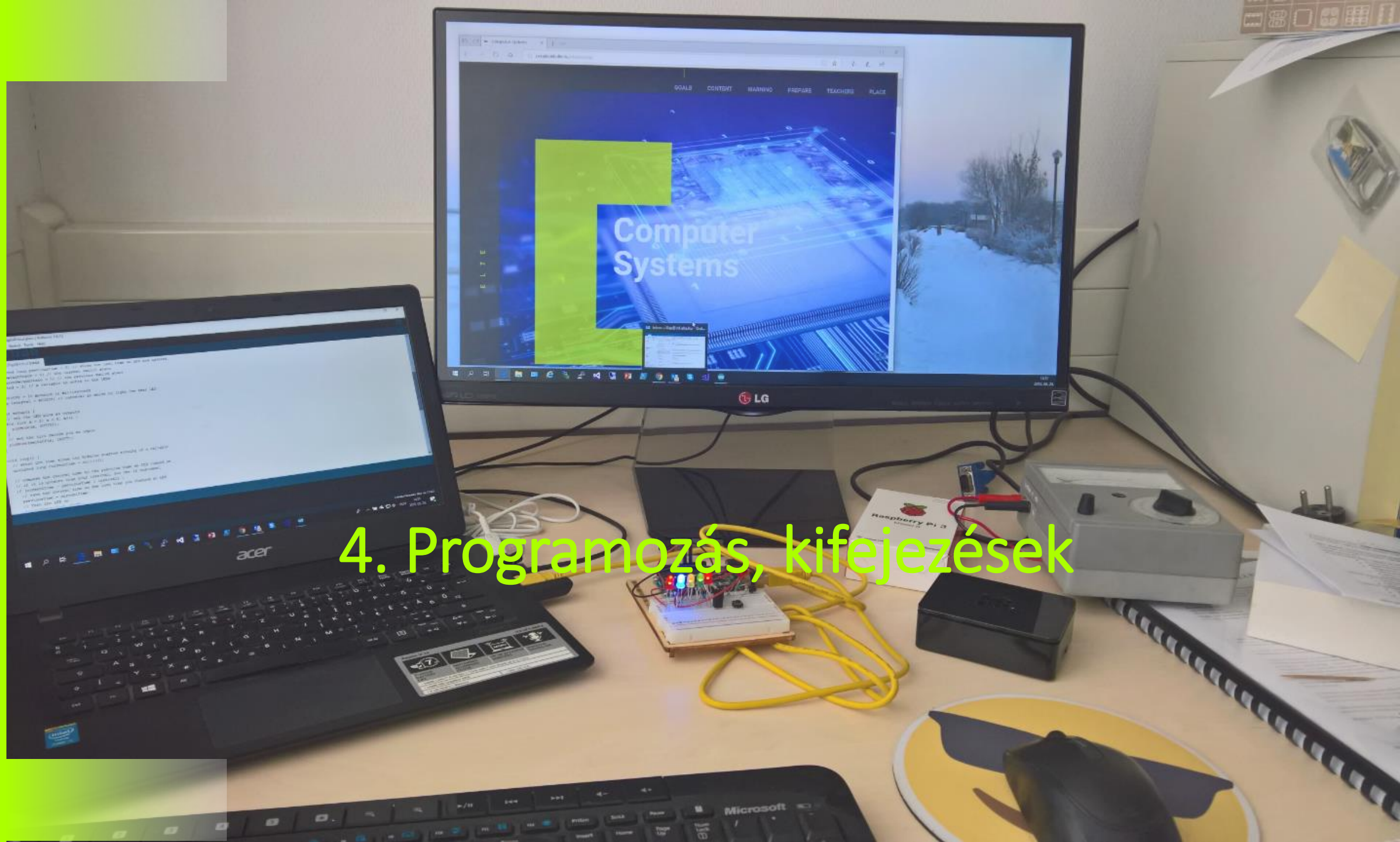


4. Programozás, kifejezések



Visszatekintés

- Számítógépek, információk, számábrázolás, kódolás
- Felépítés, operációs rendszer, kliens-szerver szerep
- Grafikus, karakteres kapcsolat, fájlrendszerek
- Alapvető parancsok, folyamatok előtérben, háttérben
- I/O átirányítás, szűrők, reguláris kifejezések
- Változó, parancs behelyettesítés
- Aritmetikai, logikai kifejezések

Mi jön ma?

- Vezérlési szerkezetek
- Elágazások
- Ciklusok
- Függvény definíció
- ...

Egy shell példa

- Mit csinál a következő script?
- alma speciális alakú:
 - X=Gyurika&y=akirugy
 - Hol használatos ez a forma?
- A beolvasott változóból kiveszi ,login' azonosítót és a jelszót (pw)!

```
read alma
login=`echo $alma|cut -f1 -d\&|cut -f2 -d=`
pw=`echo $alma|cut -f2 -d\&|cut -f2 -d=`
#
cat <<ali
Content-Type: text/html

<html>
<body bgcolor="#a1c1a1">
ali
  echo Azonosítója: $login , jelszava: $pw !
cat <<pali
</body> </html>
pali
```

Elágazás shell scriptben

- if if
 utasítások [\$x -lt 10]
 then then
 utasítások echo Kisebb mint 10
 else else
 utasítások echo Nagyobb
 fi fi
- Nem kötelező a fentiek szerinti tagolás, csak javasolt!

Feltételes parancsvégrehajtás

- if sikeres then másikparancs fi
 - sikeres && másikparancs
- if nemsikeres then másikparancs fi
 - nemsikeres || másikparancs
 - Példa:

```
$ echo szia && echo kata
```

```
szia
```

```
kata
```

```
$ hamis || echo almafa # hamis: exit 1
```

```
almafa
```

Többirányú elágazás: case

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

Elágazás példa I.

- Feladat: Olvasson be egy számot, és írja ki, hogy pozitív, vagy negatív!

```
#!/bin/sh
# ha az első sor megjegyzés, akkor az a shell
# megadása lehet
read x # beolvasás x változóba
if
    [ $x -lt 0 ]
then
    echo Az X változó negatív, értéke: $x
else
    if
        [ $x -eq 0 ]
    then
        echo A változó értéke nulla!
    else
        echo Az X változó pozitív, értéke: $x
    fi
fi
```


Elágazás példa II.

- Tetszőleges parancs is tekinthető logikai értéket adó elágazás tesztelő kifejezésnek!

```
if
  who |grep jani >/dev/null
then
  echo a jani be van jelentkezve
else
  echo a jani nincs bejelentkezve
fi
#
read x #beolvasás
case $x in
  [dD]*)      date      ;;
  [wW]*)      who       ;;
  I*|L*)      ls -l      ;; # kis l
                                   # vagy nagy L
  *)          echo rossz választás ;;
esac
```

Ciklusok shellben: FOR

- **for** változó **in** adatlista # a for ciklus általános alakja
 - **do**
 - utasítások
 - **done**
 - **for i in `who`** # a leggyakrabban használt forma
 - do
 - echo \$i
 - done
 - **for i in alma körte barack** # klasszikus for
 - do
 - echo \$i
 - done
- # sorban kiírja a „gyümölcsöket”

FOR példa - seq

- `for i in 1 2 3 4 5; do echo $i; done # „ötös” ciklus`
- Hogyan készítsünk N-szer végrehajtható ciklust?
 1. Írunk egy scriptet ami 1-től N-ig adatot produkál!
 2. Használjuk a `seq` parancsot!
- `N=10; for i in `seq $N`;do echo $i; done # 1-től 10-ig a számok`
 - `seq 2 6 # 2,3,4,5,6`
 - `seq 2 2 6 # 2,4,6`
- Bővebben: `man seq`

Bash - FOR

- `for x in $(date)` `# Bash parancsbehelyettesítés`
 - `do`
 - `cat $x`
 - `done`
- `for ((i=1;i<10;i++))` `# C stílusú for ciklus`
 - `do`
 - `echo $i`
 - `done`
- Törekedjünk a klasszikus(sh) ciklus használatra!

Ciklusok shellben: WHILE

- while utasítások # while általános alakja, ha az utolsó
do # utasítás igaz, végrehajtjuk a ciklus magot
 utasítások
done
- while
 echo -n Írd be a neved:
 read nev # a read nem ad hamis eredményt!!!
do # befejezni pl: ctrl+d, ctrl+c
 echo A Te neved: \$nev
done

Ciklusok shellben: WHILE példa I.

- Ha a read a fájl vége jelet is beolvassa, az logikai hamis lesz!
- while
 echo -n Írd be a neved:
 read nev # ekkor sorról sorra beolvassa a fájlt
 do # a fájl végére a read hamis értéket ad!
 echo A Te neved: \$nev
done <nevek.txt

Ciklusok shellben: WHILE példa II.

- Ha a paraméter fájlnev, akkor kilistázzuk annak tartalmát

```
#!/bin/sh
#
while [ $# != 0 ] # van még paraméter?
do               # igen
  if test -f $1   # $1 fájl?
  then
    echo $1 tartalma:
    cat $1
  else
    echo $1 nem fájl!
  fi
  shift # paraméterek léptetése balra
  echo Még $# paraméter maradt!
done
```

Ciklusok shellben: UNTIL

- A while igaz esetén, az until hamis esetén hajtja végre a ciklusmagot!
- until
 utasítások
do
 utasítás(ok)
done
- Több utasítás is lehet az until után, ha az utolsó hamis, akkor végrehajtja a ciklusmagot!

Ciklusok shellben: UNTIL példa

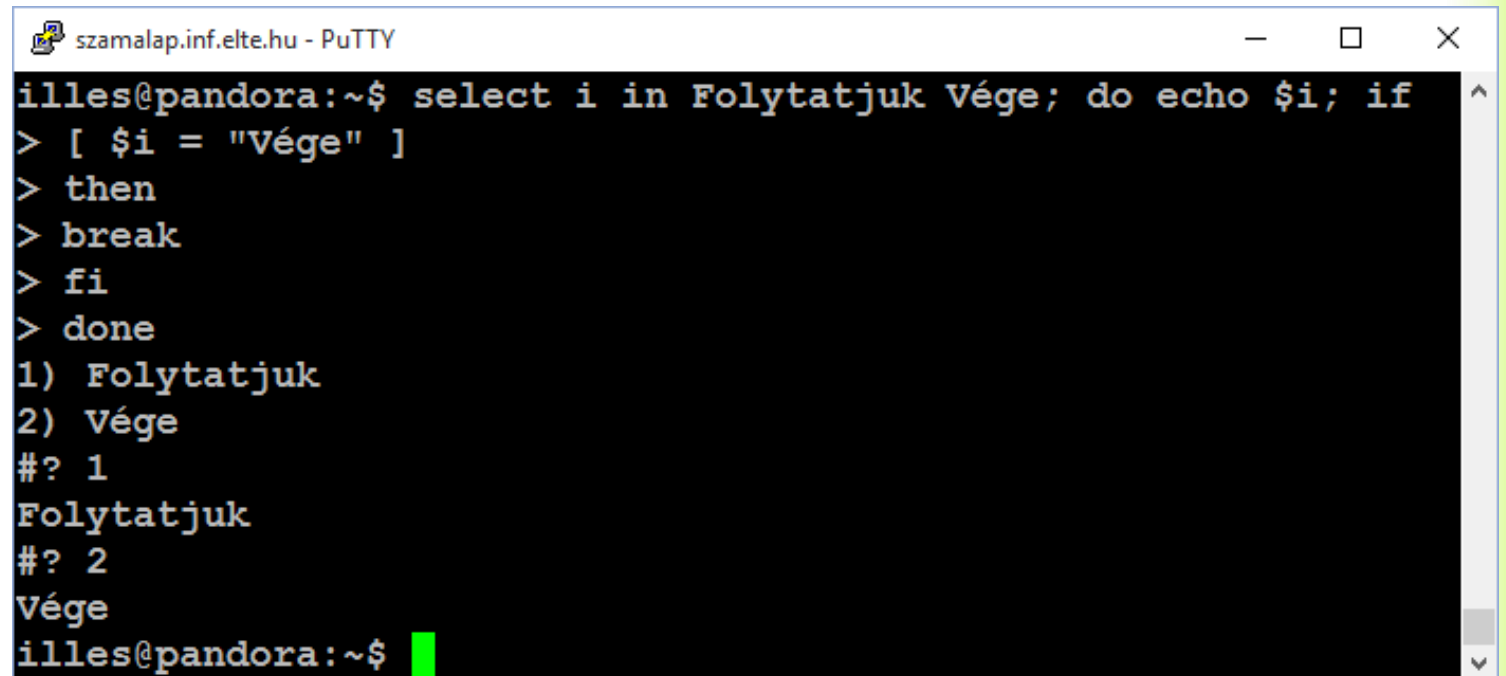
- Amíg a folytatásra nem a pontos nemet adjuk meg, addig folytatjuk a nevek fájlba írását!
- Mi lenne [\$v = [Nn]em] esetén?
 - Semmi, a test utasítás nem „bírja” a reguláris kifejezéseket!

```
#!/bin/sh
#
# kezdőérték megadása
v=igen
until
[ $v = "nem" ]
do
  echo -n Add meg a neved:
  read nev
  echo $nev >>nevek.txt
  echo Folytassuk? \(igen/nem\)
  read v
done
```

ha először nem létezik
akkor létrejön a fájl
Fontos a \
#

BASH: Select

- Menü szerkezet készítés (csak BASH)
- select i in alma barack szilva
- do
- \$i feldolgozása
- done
- Fontos: BREAK



```
szamalap.inf.elte.hu - PuTTY
illes@pandora:~$ select i in Folytatjuk Vége; do echo $i; if
> [ $i = "Vége" ]
> then
> break
> fi
> done
1) Folytatjuk
2) Vége
#? 1
Folytatjuk
#? 2
Vége
illes@pandora:~$
```

Ugró utasítások shellben

- **break**
 - Hatására az adott ciklus félbeszakad, és a done után folytatódik a végrehajtás
- **continue**
 - Hatására a ciklusmag hátralévő részén átugrik, majd folytatódik a ciklus végrehajtás.
- **exit [n]**
 - Kilép a programból és n lesz a visszatérési érték

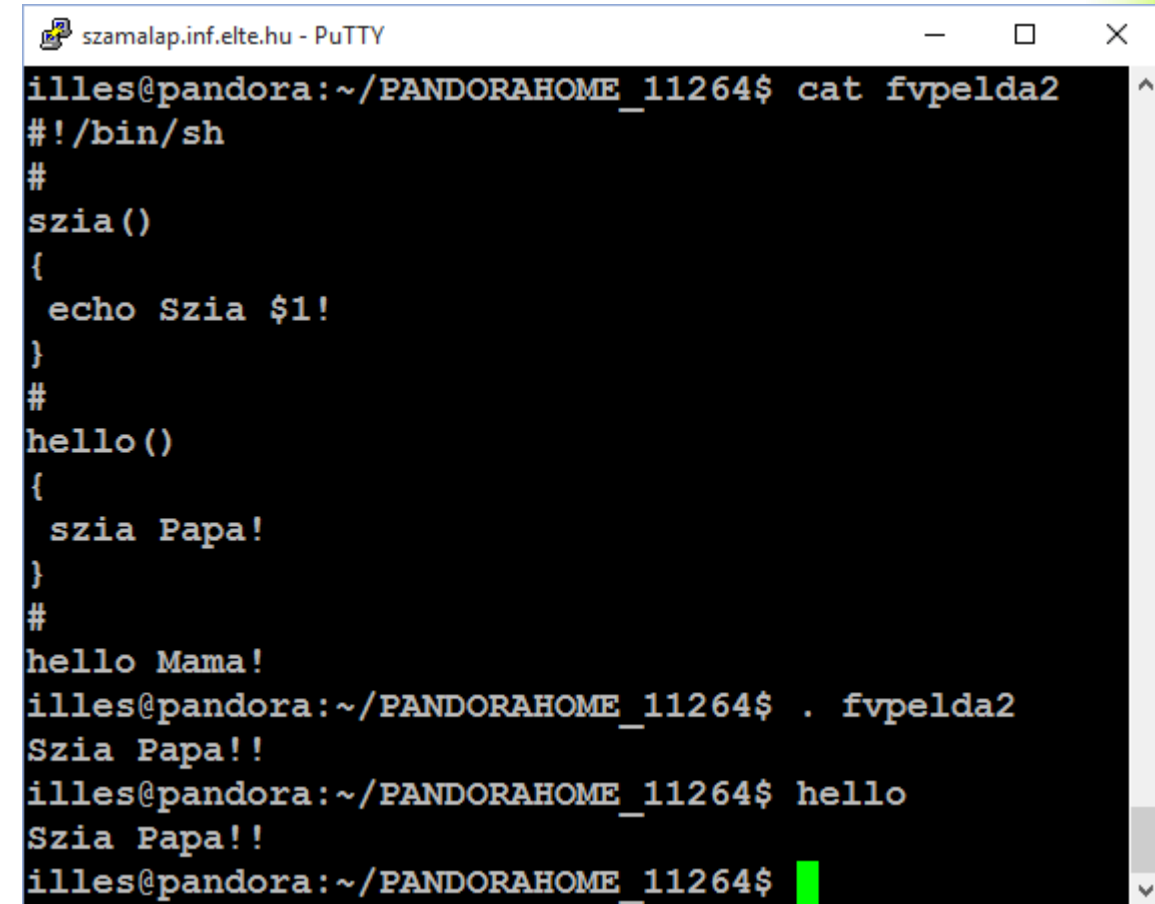
Függvény definiálás shellben I.

- C stílusú függvény, paramétereit a shell scripthez hasonlóan dolgozza fel.
- Paramétereket nem jelölhetünk!
- Függvény visszatérési értéket a return utasítással adhatunk meg!

```
illes@panda:~$ cat fvpelda
#!/bin/sh
#
szia()
{
    echo Szia $1!
}
#
szia Zoli                                #Függvény hívás
#script vége
illes@panda:~$ fvpelda
Szia Zoli!
```

Függvény definiálás shellben II.

- Egyik függvény hívhatja a másikat!
 - A hello függvény nem foglalkozik a paramétereivel!
- . scriptnév hívás: függvények elérése parancssorból.
 - unset -f fvnev # fv törlése



```
szamalap.inf.elte.hu - PuTTY
illes@pandora:~/PANDORAHOME_11264$ cat fvpelda2
#!/bin/sh
#
szia()
{
    echo Szia $1!
}
#
hello()
{
    szia Papa!
}
#
hello Mama!
illes@pandora:~/PANDORAHOME_11264$ . fvpelda2
Szia Papa!!
illes@pandora:~/PANDORAHOME_11264$ hello
Szia Papa!!
illes@pandora:~/PANDORAHOME_11264$
```

Parancs futtatási opciók

- Sh fvpelda # ha nincs futási jog, akkor is futtatja a scriptet!
- Sh -v fvpelda
 - -v kiírja a parancsot végrehajts előtt. A teljes fv-t is!
- Sh -x számolvas
 - A végrehatási szál utasításait írja ki. A -v a teljes elágazást kiírja, míg a -x a tesztelt logikai kifejezéseket írja csak.
- Sh -n számolvas
 - Szintaktikus ellenőrzés

Még több BASH

- Az alapértelmezett shell lehetőségek mellett megemlítettük a BASH megfelelőt is! PL. for ciklus
- Nem törekedtünk a BASH összes lehetőségének kiemelésére!
 - Ilyen például a tömbök használata!
- Teljes BASH leírás például innen is elérhető:
- <https://www.gnu.org/software/bash/manual/bash.html>

Script példa I.

- Mit csinál a következő példa?

```
for i
do
  case $i
  in
    [A-Z]*) F="$F $i";;
    [a-z]*) f="$f $i";;
    [0-9]*) break;;
  esac
done
echo $F
echo $f
```


Script példa II: Csomagoljunk

- Példa:

```
#!/bin/sh
#
echo # csomagoló program
echo # A szétszedés parancsa: sh ./fájlnev
# Egyenként vesszük a paramétereket
for i
do
    echo "echo $i 1>&2"
    echo "cat >$i <<'$i vege'"
    # Mivel "" a fő idézőjel, ezért ,'$i vege' is kiértékelődik!!!!
    cat $i
    echo "$i vege"      #Here input lezárása
done
```

Csomagoló használat

- A csomagoló script az eredményét a standard outputra írja!

```
$ csomagol forpelda valogat # eredmény a képernyőre  
$ ....  
$ csomagol forpelda valogat >csomag #eredmény fájlba  
$ sh ./csomag          # kicsomagolás, mivel nincs x jog  
                        # ezért így kell indítani
```

Demo...

- Csomagol ...

Köszönöm a figyelmet!

