01

Bevezetés – jogosultságkezelés, csővezeték, átirányítások

BASH script programozás

Berta Árpád

- berta@inf.u-szeged.hu
- www.inf.u-szeged.hu/~berta
- Irinyi magasföldszint, Mesterséges Intelligencia kutatócsoport, 46-os szoba

- Fogadó óra: szerda 14:00-15:00-ig (de pénteken is megtaláltok, írjatok e-mailt)

- Belső mellék: 3014

Követelmények

- Zárhelyi dolgozat időpontja 2015. május 9.
- 40 pont szerezhető

  [0-20) pont: elégtelen (1)

  [20-25) pont: elégséges (2)

  [25-30) pont: közepes (3)

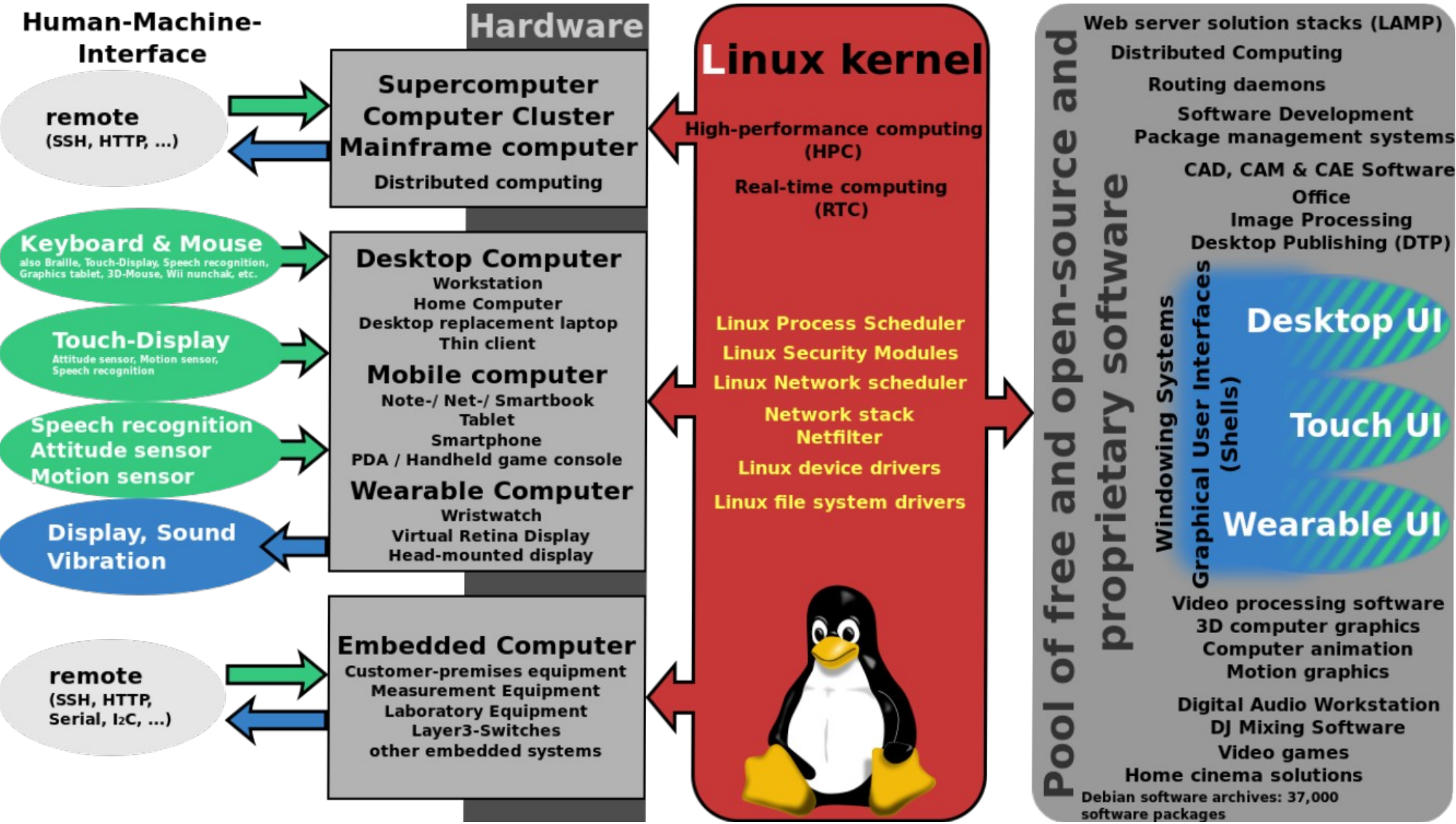
  [30-35) pont: jó (4)

  [35-40) pont: jeles (5)
- Pótolni nem lehet, csak javítani az elégtelent elégségesre a vizsgaidőszak első hetében kijelölt időpontban.

Előfeltétel

- progalapon megismert alap parancsok
- jegyzetben kiadott alap parancsok

Felépítés - GNU/Linux



GNU/Linux - Felépítés

- 1. hardver
- 2. kernel:
  - Az operációs rendszer lényegi része.
  - Feladata az erőforrások (memória, processzor, háttértár, perifériák) kezelése, felügyelete és kiosztása, a programok futtatása, az állományrendszer karbantartása, stb.
- 3. shell:
  - hozzáférést biztosít a kernel funkcióihoz, kényelmi szolgáltatások, programok indítása: **parancsértelmező**
  - egy shell a BASH (Bourne Again Shell), ezt vizsgáljuk a félév során
- 4. alkalmazások: mindenféle egyéb program

Parancsok

- PARANCS -egybetűskapcsoló (vagy --hosszúnevűkapcsoló) PARAMÉTER1, PARAMÉTER2, ….
- Például:
  - ls -l /home/
  - cd /var/
  - man - -help
  - exit

Állományrendszer

- Fa felépítésű, minden állomány a / gyökérből indulva megadható
- A UNIX állományok típusa:
  - közönséges fájl
  - speciális: meghatározott szerkezetű, különleges célú
    - könyvtár (directory)
    - eszköz (device)
    - szimbolikus link (symbolic link)
    - nevesített FIFO cső (named pipe, FIFO)
    - kommunikációs végpont (socket)
- Elérési utak:
  - Abszolút: / gyökértől indulva megadott (~ is abszolút)
  - Relatív: . jelenlegi könyvtárhoz képest megadott (.. is relatív)

Állományrendszer (man 7 hier)

- /boot: az operációs rendszer elindulásához szükséges
- /bin, /sbin, /usr/bin, /usr/sbin: futtatható állományok gyűjtőhelye
- /dev: eszközállományokat tartalmaz (terminálok, stdin/out/err, ram)
- /etc: adminisztrációs, konfigurációs állományok, kritikus beállítások (/etc/fstab)
- /home: a felhasználói könyvtárakat tartalmazza
- /lib: programok által használt függvénykönyvárakat tartalmaz
- /mnt, /media: külső állományrendszerek gyűjtőhelye
- /opt, /var: vegyes beállítások, adatok, programok (/var/www)
- /root: rendszergazda home könyvtára
- /tmp: ideiglenesen létrehozott állományok
- /usr: felhasználók által elérhető közös adatok, információk, programok

Állományrendszer fizikai felépítése

1. Boot block (nulladik blokk): az ebben levő rövid program tölti be a rendszert
2. Superblock (első blokk): az állományrendszer részleteit és a belső táblák adatait tartalmazza
3. Inode tábla: az inode-ok adatait tartalmazza

- Az inode (index node)
  - egy adott állomány minden fontos adatát tartalmazza: méretet, típust, tulajdonost, a hozzáférési jogokat, a háromféle dátumot (access: elérés, modofy: modósítás, change: inode változás), az állományhoz tartozó lemezblokkok sorszámait, valamint a merev láncok számát avagy a láncszámot.
  - Szigorúan véve az inode-okat azonosíthatjuk az állományokkal.
    - stat, (touch)
  - Minden inode egyedi sorszámot kap.
- Minden könyvtárhoz tartozik egy állomány. Ez a speciális állomány tartalmazza a könyvtárban levő állományok nevét és inode-számát.

Jogosultságkezelés

- minden állománynak van:

  (-R kapcsoló: rekurzívan az összes almappára végrehajtja)

  - tulajdonosa (chown),
  - tulajdonos csoportja (chown, chgrp),
  - jogosultsági szintje (chmod)
- háromféle jogosultsági szintet különböztethetünk meg:
  - tulajdonos jogosultsága (u)
  - csoport jogosultsága (g)
  - mindenki más jogosultsállományága (o)könyvtár
- háromféle jogosultsági módot különböztetünk meg:

	állomány	könyvtár
olvasási (r)	olvasható	listázható
írási (w)	módosítható	állományok hozhatók létre, vagy törölhetők
végrehajtási (x)	futtatható	be lehet lépni

Jogosultság numerikus alak

- 0 → semmilyen jogosultság nincs
- **1** → **x** (végrehajtási jog)
- **2** → **w** (írási jog)
- 3 → wx
- **4** → **r** (olvasási jog)
- 5 → rx
- 6 → rw
- 7 → rwx

chmod példák

- abszolút megadás:
  - chmod 755 filename
  - chmod a=rwx filename
- relatív használat:
  - chmod +rwx filename (= chmod ugo+rwx filename)
  - chmod -rwx filename (= chmod a-rwx filename)
  - chmod ug-w filename

Feladat: nézzük meg a különböző jogosultsági eseteket állományok és könyvtárak esetében

Linkelés

- egy állományt több helyen/néven is elérhetővé tegyünk a fájlrendszerben
- Hard link:
  - ln TARGET LINKNAME
  - újra létrehozza a fájlt, ugyanarra az inodera mutat, bármelyik változik akkor egyszerre változik az összes
  - eredeti törlése esetén megmarad a hard link
- Soft (v. symbolic) link:
  - ln -s TARGET LINKNAME
  - hagyományos értelemben egyszerű link (tükör) jön létre, ami az eredeti állományra mutat
  - eredeti törlése esetén megmarad a soft link, de sérül és nem működik többet

Csatornák (streamek)

- 3 standard be- és kimeneti csatorna létezik:
  - Stdin (0) : alapértelmezetten a billentyűzet
  - Stdout (1) : alapértelmezetten a képernyő (terminál)
  - Stderr (2) : alapértelmezetten a képernyő (terminál)
- Ezek átirányíthatóak más eszközökre (device) vagy fájlba

Streamek átirányítás

- < ÁLLOMÁNY: stdin átirányítása (a megadott fájlból olvas)
- > ÁLLOMÁNY: stdout átirányítása (a megadott fájlba ír, a létező állomány felülírásával)
- >> ÁLLOMÁNY: stdout átirányítása (a megadott fájlba ír, a létező állomány végéhez való hozzáfűzéssel)
- 2> ÁLLOMÁNY: stderr átirányítása (a megadott fájlba írja a hibaüzeneteket)
- &> ÁLLOMÁNY: stdout és stderr átirányítása ugyanabba a fájlba

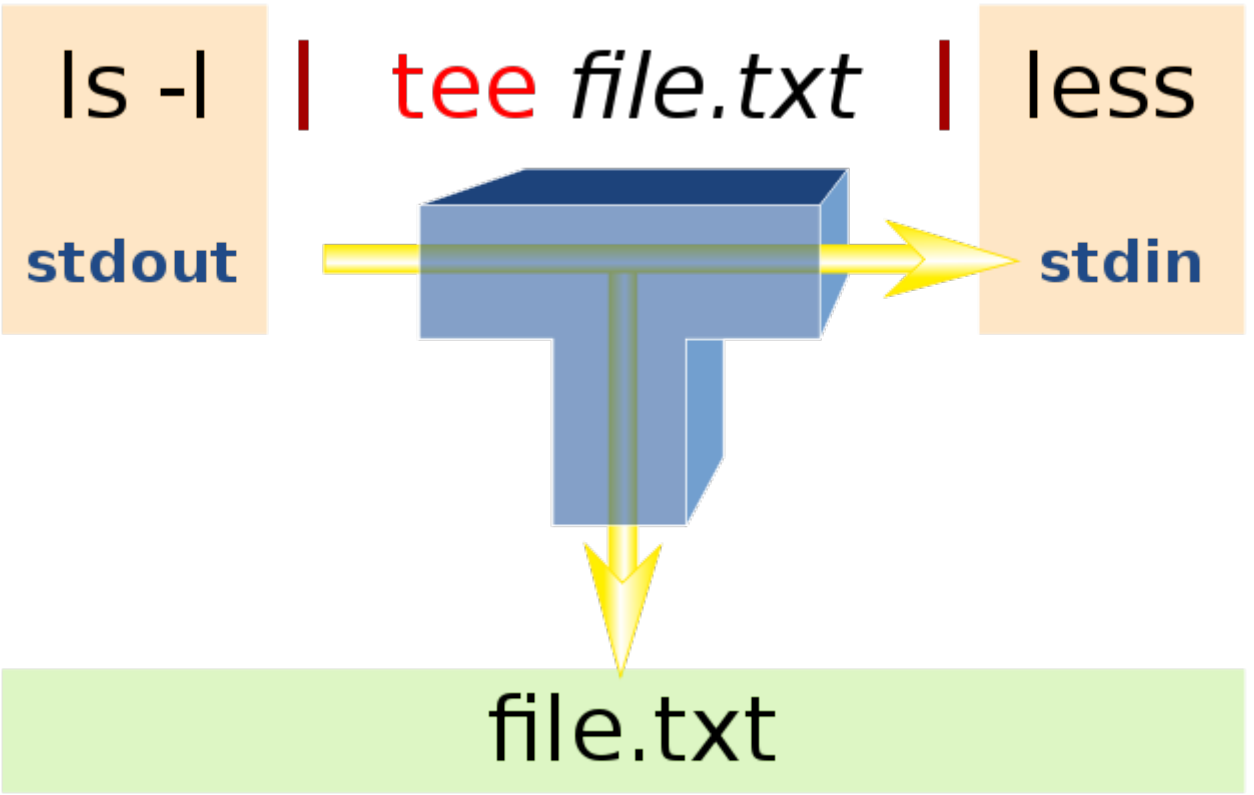
Feladat: Stdinről olvasott tartalmat irányítsuk át egy fáljba, hozzunk létre belőle kétféle linket, majd a fájl tartalmát irányítsuk át egy másik terminálba

Csővezeték (pipeline)

- |
- Egyik parancs kimenetét (stdout) irányítjuk a következő parancs bementének (stdin)
- Feladat:
  - Írjuk ki a feladat.dat fájl 3.-9. sorát (head és tail felhasználásával)

Szétbontás - tee

- stanrd kimenetre és a paraméterként adott fájl(ok)ba irányít
- - a: append



BASH script

- script kötegelt végrehajtása:
  - a parancsértelemező nem a parancssorból, hanem a fileból olvassa soronként a parancsokat
- saját parancsot hozhatunk létre (automatizálás)
- felépítése:
  - #! /bin/bash első sor, parancsértelmező fejrész
  - lehetőség van bármilyen parancs (akár program) végrehajtására
  - soronként egy parancs (de lehet többsoros parancs is sorvégi: \)
  - változók, vezérlési szerkezetek (if, for, while, …)
  - !!!nagyon kötött a szintaxis!!!

BASH script futattása

- bash SCRIPTNAME PARAMÉTEREK
  - ha nem adtunk meg parancsértelmező fejlécet, futtatáskor kell megadnunk, hogy miben fusson le
- futtatás egyszerű futtatható állományként:
  - kell a parancsértelmező fejléc (hibás futtatás elkerülése végett)
  - chmod +x SCRIPTNAME
  - ./SCRIPTNAME PARAMÉTEREK

BASH script alapok

- Egysoros kommentelés: #
- 'SZÖVEG': bármilyen karaktert tartalmazhat a SZÖVEG – nincs olyan, ami spec karakterként viselkedne (nem lehet változóra hivatkozni)
- ”SZÖVEG”: csak alap karaktereket tartalmazhat + spec karakterek, amik megőrzik jelentésüket: változóhívás, parancs behelyettesítés, ...

Parancsmegadás

- szokásos módon parancs megadása (mint ahogy a parancssorban megszokták) soronként
- beágyazott parancs megadása:

  ekkor a parancs lefut és az eredménye fog behelyettesítődni (parancsként megjelenni) az adott sorban

  - `PARANCS`
  - **$(**PARANCS**)**

Változók

  - lokális script változók (csak az adott alshellben érhető el)
  - környezeti változók (öröklődik a további alshellekbe)
- értékadás: VALTOZO_NEV=érték
  (!!nincs szóköz az = után !!!)
- hivatkozás: $VALTOZO_NEV (vagy ${VALTOZO_NEV})
- beolvasás standard bemenetről: read VALTOZO_NEV
- névadás: betűket, számokat és _ jelet tartalmazhat, de nem kezdődhet számmal, hagyomány szerint nagybetűsek
- érték: alapesetben szövegként kezel minden változót

Környezeti változók

- környezet:
  - név – érték párok halmaza
- (globális) shell szintű változók
- env, printenv: környezeti változók listázása
- export VALTOZO_NEV vagy export VALTOZO_NEV=érték:
  - környezeti változó létrehozása
- unset VALTOZO_NEV: környezeti változó feloldása

Paraméterek

- hivatkozás: $1, $2, …. , $9, ${10}, ${11}, ….
- összes paraméter listában: $*
- a script neve: $0
- paraméterek száma: $#

Feladat: írjunk egy olyan scriptet, ami kiírja az első és a harmadik paramétert, majd kiírja az összeset.

Matematikai műveletek

- **$((**KIF**))**
  - csak egész számokon értelmezett, eredmény is egész (használhatóak változók)
  - +, -, *, /, %, ++, - –, **(hatványozás), !, ~, &, ^, &&, ||, ?:, =, +=, -=, *=, /=
  - < <= > >= == != (igaz: 1, hamis: 0)

Logikai műveletek

if [FELTÉTEL **]; then**

PARANCS(OK)

elif [FELTÉTEL **]; then**

PARANCS(OK)

else

PARANCS(OK)

fi

!! [ ] zárójel előtt, után KELL a SZÓKÖZ !!

!! FELTÉTEL megadása csak speciális kapcsolókkal lehetséges !! Hagyományos módon átirányítás fog történni !!

Logikai műveletek

if ((FELTÉTEL **)); then**

PARANCS(OK)

elif ((FELTÉTEL **)); then**

PARANCS(OK)

else

PARANCS(OK)

fi

Feltételes kifejezések

- Numerikus összehasonlítás:
  - KIF1 -eq KIF2 (egyenlő)
  - KIF1 -ne KIF2 (nem egyenlő)
  - KIF1 -lt KIF2 (kisebb mint)
  - KIF1 -le KIF2 (kisebb egyenlő)
  - KIF1 -gt KIF2 (nagyobb mint)
  - KIF1 -ge KIF2 (nagyobb egyenlő)
- Logikai kifejezések:
  - KIF1 -a KIF2 (and) vagy KIF1 && KIF2
  - KIF1 -o KIF2 (or) vagy KIF1 || KIF2
  - !KIF1 (tagadás)
- Csoportosítás zárójelekkel megtehető: (KIF)

Feltételes kifejezések

- Szöveges összehasonlítás (egyetlen szó)
  - KIF1 == KIF2
  - KIF1 != KIF2
  - -z KIF (üres szó)
  - -n KIF (nem üres szó)
- Állományok jellemzőinek a vizsgálata:
  - -e KIF (létezik az állomány), -d KIF (létezik a könyvtár), -f KIF (létezik a közönséges állomány), -h (létezik a szimbolikus link), -p (létezik a csővezeték)
  - -r, -w, -x KIF olvasási, írási és végrehajtási jog az aktuális felhasználó szemszögéből
  - -O, -G KIF tulajdonos, csoport megegyezik-e az aktuális felhasználóval
  - -s KIF (nem üres)
  - KIF1 -nt KIF2 (újabb mint)
  - KIF1 -ot KIF2 (régebbi mint)
  - KIF1 -ef KIF2 (azonos a két állomány)

for

- lista bejárása:

  for VÁLTOZÓ **in** LISTA**; do**

  PARANCS(OK)

  done

- LISTA megadása:
  - szöveg felsorolva szóközzel elválasztva
  - mintaillesztéssel fájlok listája
  - $() ← beágyazott parancs kimenete (ami szóközes vagy több soros)
  - változóhivatkozás (olyan változó, ami szóközes vagy többsoros tartalommal rendelkezik)

for

- index alapú for:

```
for (( i=0; i<=$N; i++ )); do
    PARANCS(OK)
done
```

while, egyebek

while [FELTÉTEL **]; do**

PARANCS(OK)

done

- vagy másképpen

while ((FELTÉTEL **)); do**

  - PARANCS(OK)

done

- ehhez hasonlóan létezik **until** vezérlési szerkezet
- alkalmazható a **break**, **continue** és az **exit**

függvények, függvényhívás

```
function FUGGVENYNEV() {
    PARANCS(OK)
}
```

FUGGVENYNEV PARAM1 PARAM2 … PARAMN

Feladatok

- Írjunk olyan scriptet, ami a paraméterül kapott számokat összeadja és végül egy osszeg.txt fájlba írja a kapott eredményt
- Írjunk olyan scriptet, ami a paraméterül kapott számig kiírja 0-tól indulva a páros számokat.
- Nézzük meg a fájlokra vonatkozó feltételes kifejezéseket. Írjunk egy scriptet, amiben a paraméterül kapott mappában lévő fájlokat tudjuk vizsgálni. Például kiírjuk a legújabb fájlt.