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, 45/A szoba
- Fogadó óra: e-mailes egyeztetés alapján
- Belső mellék: 6714

Tematika

- 6. hét, 2017.03.17. (P) 10:00-12:00 IR223
  - Bevezetés – jogosultságkezelés, csővezeték, átirányítások
  - BASH script programozás
- 13. hét, 2017.05.05. (P) 12:00-14:00 IR224
  - Szűrők, Reguláris kifejezések
  - AWK

  14. hét, 2017.05.14. (V) 23:59
  - otthoni beadandó feladatok, **beadási határidő** (10 pont)
- 15. hét, 2017.05.19. (P) 14:00-16:00 IR224
  - **zh** (40 pont)

Követelmények

- beadandó+zh, összesen 50 pont szerezhető, de

  0-19 pont: elégtelen (1)

  20-24 pont: elégséges (2)

  25-29 pont: közepes (3)

  30-34 pont: jó (4)

  35- pont: jeles (5)

- a javító zh-n viszont már 50% kell a (2) érdemjegyhez (nem számítanak bele a félév során szerzett pontok)

Előzmény

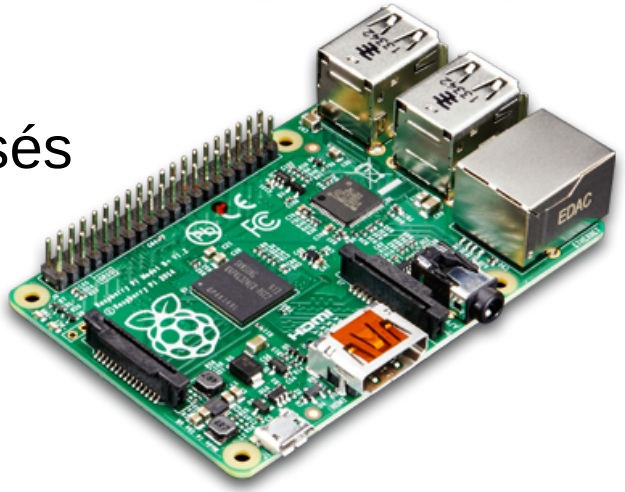
- programozás alapja gyakorlat, linuxos bevezető
- akinek ez nem volt, vagy már nem emlékszik, az nézze át Griechisch Erika gyakorlati jegyzetének **első 4 oldalát** a mintaillesztésig
- az alap parancsokkal itt már nem foglalkozunk

Tanuláshoz mit használj?

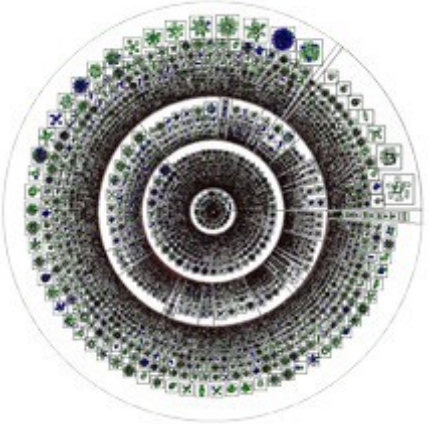
- Valamilyen linux disztribúciód van? OK
- Nem linuxod van, hanem valami más?
  - ssh a h-s azonosítóddal az Irinyi kabinetbe (linux.inf.u-szeged.hu) (legkönnyebb, viszont nincs GUI)
  - VirtualBox, vmware: virtuális linux, egy iso alapján (könnyű, viszont erőforrásigényes)
  - Live USB linux használata, egy iso alapján. Pl egy pendrivera telepítve (könnyebb, de nem garantált a gyakorlási fájlok „túlélése”)
  - telepíthetsz mellé valamilyen linux disztribúciót (nehezebb, több idő)
  - Windows 10 és OSX esetében is elérhető olyan beállítás, amely segítségével elérhető a bash shell (de ennek elérése nehézkes és hasonlósága a benti rendszerekhez nem garantált, csak saját felelősségre)

Motiváció

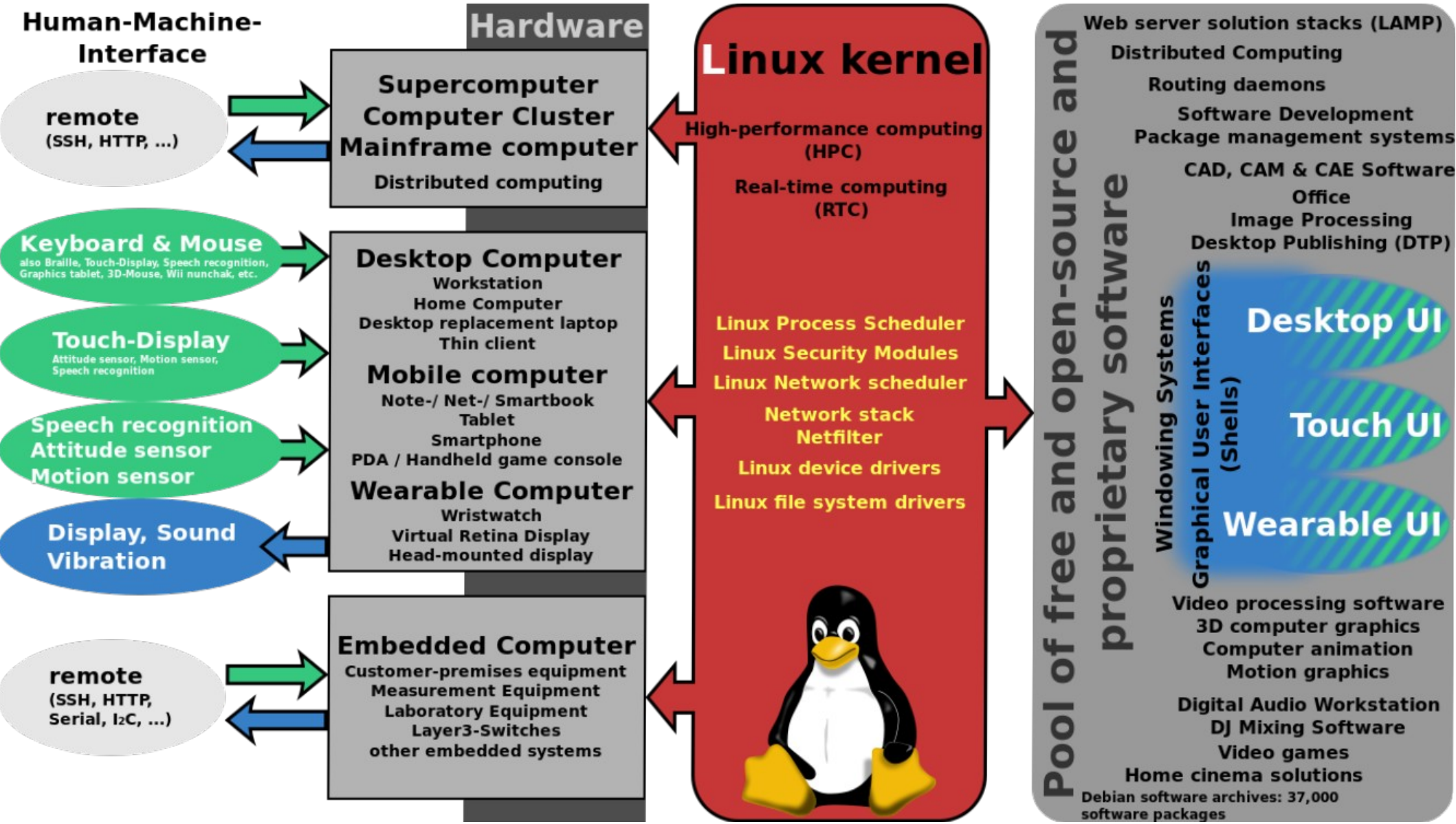
- a programozó munkáját segítő eszközök:
  - fájlrendszer kezelése
  - adattisztítás
  - programkód szöveges tartalmában való keresés
  - statisztikák készítése
  - ...



- szerver adminisztráció
- kisteljesítményű eszközök/szenzorok (raspberry pi)
- rálátás egy operációs rendszer belső logikájára/működésére

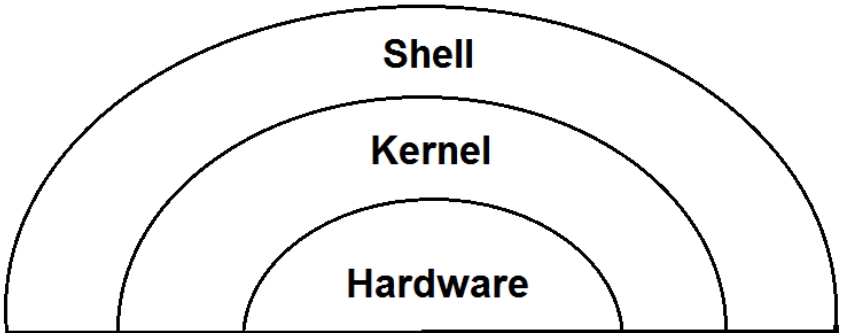


Felépítés - GNU/Linux



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



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

Parancsok

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

  Feladat hajtsuk végre az alábbi parancsokat:
  - chmod +x touch.sh
  - ./touch.sh

Mintaillesztés

- Használata: állományok nevének a megadására
- Jelölő karakterek:
  - * tetszőleges karakterből álló szó
  - ? a helyén valamilyen tetszőleges karakter
  - [HALMAZ] a felsorolt karakterek valamelyike
  - [ELSŐ-UTOLSÓ] hasonlóan az előzőhöz csak egy tól-iggel megadva
  - [^HALMAZ] a fel nem sorolt karakterek lehetnek ott

Jogosultságkezelés

- minden állománynak van:

  (-R kapcsoló: rekurzívan az összes almappára végrehajtja)

  - felhasználó tulajdonosa – user owner (chown),
  - csoport tulajdonosa – group owner (chgrp), (egy felhasználó akár több csoportnak is lehet a tagja)
  - jogosultsági szintje (chmod)
- háromféle jogosultsági szintet különböztethetünk meg:
  - felhasználó tulajdonosra értelmezett (u)
  - csoport tulajdonosra értelmezett (g)
  - mindenki másra értelmezett (o)
- 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 **felülírás**ával)
- >> ÁLLOMÁNY: stdout átirányítása (a megadott fájlba ír, a létező állomány végéhez való **hozzáfűzés**sel)
- 2> ÁLLOMÁNY: stderr átirányítása (a megadott fájlba írja a **hibaüzenet**eket)
- &> ÁLLOMÁNY: stdout és stderr átirányítása ugyanabba a fájlba
- 2>&1: a stderr-t ugyanoda irányítja, ahová a stdout irányítva lett
- 1>&2: a stdout-ot ugyanoda irányítja, ahová a stderr irányítva lett

Feladat:

- `mkdir workspace` parancs hibakimenetének átirányítása egy err.log fájlba
- Stdinről olvasott tartalmat irányítsuk át egy fáljba, hozzunk létre belőle kétféle linket, majd a fájl tartalmát irányítsuk át egy másik terminálba

Kiíratás

- cat – kiírja a fájl(ok) tartalmát
- less – lapozhatóan írja ki a fájl(ok) tartalmát
- wc – kiírja a fájlban található sorok (1), szavak (2), bájtok (3) számát
- head – első tíz sort írja ki
- tail – utolsó tíz sort írja ki
- sort – a sorokat abc sorrendbe rendezetten írja ki a sorkezdő karakter alapján (de pl számok alapján is rendez -n kapcsolóval)
- uniq – egymás után ismétlődő sorokat egyszer írja ki

Csővezeték (pipeline)

- |
- Egyik parancs kimenetét (stdout) irányítjuk a következő parancs bemenetének (stdin)
- Átirányítással három sorban oldható meg ugyanez:

```
sort -n feladat02.dat > feladat02.sort
uniq < feladat02.sort
rm feladat02.sort
```

  (ha hosszabb a csővezeték, akkor még több sor kell)
- Ugyanez csővezetékkel egyszerűbben, egysorban:

```
sort -n feladat02.dat | uniq
```

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: \
    - lehet egy sorban több „sornyi” parancs is: ; (sortörés szimbólum)
  - változók, vezérlési szerkezetek (if, for, while, …)
  - !!!nagyon kötött a szintaxis!!!

BASH script alapok

- Parancsértelmező fejrész: #! /bin/bash
- Egysoros kommentelés: #
- 'SZÖVEG':
  - nincs jelentése a spec karaktereknek
  - pl nem lehet változóra hivatkozni, nincs parancsbehelyettesítés
- ”SZÖVEG”: jelentése van a spec karaktereknek:
  - $ változónév hivatkozás
  - ` ` közé írt parancs végrehajtódik és az eredmény behelyettesítődik
  - \ kezdve a fenti spec karakter is kiírható, pl: \$

Változók

- 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: **szöveg**

  Feladat: Kérjünk be egy szöveget, adjuk át egy változóba majd írassuk ki

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

Parancsmegadás

- szokásos módon parancs megadása (mint ahogy a parancssorban megszoktuk)
  - ekkor végrehajtódik és a STDOUT/ERR-ra íródik a kimenet
- hivatkozás a parancs eredményére:

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

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

Feladat: ls kimenetét adjuk át egy változónak és írassuk ki

Matematikai jelölés

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

Összefoglaló megjegyzés

- Jelölésbeli különbség van parancs végrehajtása és a parancs kimenetére való **hivatkozás** között

  PARANCS ↔ $(PARANCS)

- Jelölésbeli különbség van változó értékadás és változó értékére való **hivatkozás** között

  VARIABLE=10 ↔ $VARIABLE (vagy ${VARIABLE})

- Jelölésbeli különbség van matematikai számítás elvégzése és ugyanennek az eredményére való **hivatkozás** között

  ((3+2)) ↔ $((3+2))

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 sorba az összes paramétert
- a következő sorban pedig összeadja az második és harmadik paramétert

Logikai műveletek (1.)

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 !!

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)
  - KIF1 -o KIF2 (or)
  - !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 fájl)
  - KIF1 -nt KIF2 (újabb mint)
  - KIF1 -ot KIF2 (régebbi mint)
  - KIF1 -ef KIF2 (azonos a két állomány)

Logikai műveletek (2.)

if ((FELTÉTEL **)); then**

PARANCS(OK)

elif ((FELTÉTEL **)); then**

PARANCS(OK)

else

PARANCS(OK)

fi

Logikai műveletek (3.)

if PARANCS**; then**

PARANCS(OK)

fi

- a PARANCS kilépési állapota alapján lesz igaz vagy hamis.
- általában a kilépési állapot:
  - exit 0 ha minden OK (if esetén true)
  - exit 1 valamilyen hiba történt (if esetén false)
- Az exit status 0 – 255 vehet fel értéket, tehát van lehetőség saját hiba kilépési státusz használatára is (kivételkezelés)

PARANCS

if (($? == 0)); then

#Ahol a $? az előző PARANCS exit állapotára hivatkozó változó.

Logikai műveletek (4.)

```
case $VÁLTOZÓ in
    minta1)
        PARANCS(OK);;
    minta2|minta3)
        PARANCS(OK);;
    *)
        PARANCS(OK)
esac
```

Feladat

- Az első két paraméterként kapott szám közül írjuk ki a maximális értékűt (kétféle if jelöléssel próbáljuk ki).

  Megoldás: 05_01.sh

for (lista alapú)

- lista bejárása:

  for VÁLTOZÓ **in** LISTA**; do**

  PARANCS(OK)

  done

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

for (index alapú)

- index alapú for:

  for ((i=0; i<=$N; i++))**; do**

  PARANCS(OK)

  done

- ritkábban fogjuk használni

while, egyebek

while [FELTÉTEL **]; do**

PARANCS(OK)

done

- vagy másképpen

  while ((FELTÉTEL **)); do**

  - PARANCS(OK)

  done

- ehhez hasonlóan létezik **until** vezérlési szerkezet

Tetszőleges ciklusnál használható: a **break**, **continue** és az **exit**

függvények, függvényhívás

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

FUGGVENYNEV PARAM1 PARAM2 … PARAMN

Feladatok

- Írjunk olyan scriptet, a paraméterként megadott állományok közül összeszámolja a közönséges állományokat és végül egy eredmeny.txt fájlba írja a kapott eredményt

  Megoldás: 05_02.sh

- Írjunk egy scriptet, amiben a paraméterül kapott mappában lévő fájlok közül kiírja a legújabb fájlt.
  Megoldás: 05_03.sh