

# SzámRend – kérdezz-felelek

1. **Milyen jellemző paraméterei vannak egy mai asztali számítógépnek?**  
Processzor, memória, háttértár, alaplap tulajdonságai, hűtés típusa stb.
2. **Mi a cache szerepe a mikroprocesszorban?**  
Cache: gyorsítótár. Különböző órajelű egységek sebessékgülönbségeinek kompenzációja.  
Pl.: gyorsabb órajelű egység küld adatot lassabbra: az adatok átmenetileg a cache-ben tárolódnak, így a gyorsabb egységek nem kell a lassabbra várni.
3. **Milyen a kettes komplexumos számábrázolás?**  
Fixpontos számábrázolási módszer. A pozitív számokat hagyományos módon alakítjuk át kettes számrendszerbe, míg a negatívokat a konverzió után bitenként negáljuk, majd hozzáadunk 1-et.
4. **Mit tud az UTF-8 kódolásról? Mire jó?**  
[8-bit Unicode Transformation Format] Változó hosszúságú UNICODE karakterkódolási eljárás. Bármilyen UNICODE karaktert képes reprezentálni, ugyanakkor visszafelé kompatibilis a 7 bites ASCII szabvánnyal.
5. **Soroljon fel legalább 3 memóriatípust!**
  - RAM
  - ROM
  - Flash
  - Cache
6. **Soroljon fel olyan hétköznapi eszközöket, amelyekben „számítógép” van!**  
Okostelefon, okosóra, légkondicionálók, autók, házimozi-hifi rendszerek, közlekedési lámpák stb.
7. **Mi a különbség a szerver és egy kliensgép között?**  
Szerver: nagy teljesítményű hardware vagy software, ami a következőket biztosít(hat)ja más számítógépek (kliensgépek) számára egy hálózaton keresztül:
  - tárolt adatok elérése
  - szerver által előállított adatok elérése
  - a szerver hardware erőforrásainak kihasználása (pl. nyomtató, processzor)
  - egyéb szolgáltatások eléréseKliens: hálózaton keresztül kapcsolódik a szervergéphez, igénybe veszi annak szolgáltatásait.
8. **Soroljon fel legalább 3 operációs rendszert!**  
Windows, Unix, Linux (+ disztrók), MacOS, iOS, Android, DOS stb.
9. **Mi a különbség az SSH és a Telnet kapcsolat között?**  
Az SSH kapcsolat biztonságos (titkosított) kommunikációs csatornát épít ki a kliens és a szerver között, titkosított adatátvitellel dolgozik. (A Telnet nem titkosít...)

**10. Milyen szolgáltatásai vannak egy mai operációsrendszernek?**

Programok futtatása, be-/kivitel, szerkeszthető fájlrendszer, hibakeresés/-kezelés, erőforrás elosztás, védelem, hálózati kapcsolat, kommunikáció, kliens-szerver megoldások, közös és osztott háttértárak, több felhasználói fiók stb.

**11. Mit ért Shell alatt? Nevezzen meg legalább kettőt!**

Rendszerhéj: felület, amely segítségével a felhasználó kommunikálhat a kernellel (rendszermag). Parancssori: Bash, POSIX shell, PowerShell, Command Prompt. Grafikus: (nem vettük, de pl Windows meg minden, ami nem ~~~ DOS)

**12. Mi az alias? Hol találkozott vele?**

Shell parancs: egy szó helyettesítése egy másikkal. Többnyire parancsok helyettesítésére. pl. PowerShell

**13. Milyen UNIX fájlrendszer jellemzőket tud megemlíteni?**

A fájlrendszer kiindulópontja a „root” („/”); a fájlok és könyvtárak rendszere fát alkot; hierarchikus szerkezetű; kétféle bejegyzés: könyvtár, fájl; eszközök is rendelkeznek fájlnévvel; linkelés: fájlra (vagy annak tartalmára) mutató bejegyzés

**14. Milyen fájlrendszeret ismer?**

FAT32, exFAT, NTFS, ext2, ext3, ext4.

**15. Milyen fájlnév konvenciókat ismer UNIX-ban?**

Kisbetűs, ékezet nélküli elnevezések; elválasztásnak „-” vagy „\_”; rejtett fájlok: „.”-tal kezdődnek; különleges karakterek használhatóak, de nem javallottak

**16. Milyen fájl jellemzőket ismer UNIX-ban?**

A fájl a rendszerben egy strukturálatlan bájt-sorozat (nincs fájlvég-jel). A rendszer nyilvántartja a fájl hosszát, utolsó módosítás időpontját, a fájlrendszer hány pontjáról hivatkozunk erre a fájlra stb.

**17. Magyarázza el a UNIX-ban lévő alapvető hozzáférési jogosultsági rendszert!**

Minden könyvtárbejegyzés vagy fájl (különböző típus), vagy könyvtár. minden bejegyzés jogosultságai 3 csoportot alkotnak: a tulajdonos/felhasználó jogosultságai (u), a csoport jogosultságai (g), bárki/mindenki jogosultságai (o). minden csoportnak 3 féle jogosultsága lehet: olvasási (r), írási (w), futtatási (x). Módosítás: chmod parancs.

**18. Milyen kiegészítő jogokat ismer UNIX-ban?**

setuid: a futtatható állomány a fájl tulajdonosának jogosultságaival fut.

setgid: a futtatható állomány a fájl csoportjának jogosultságaival fut.

sticky bit: a könyvtár fájljai csak a tulajdonos által módosíthatóak.

**19. Milyen célt szolgál a UNIX-folyamatok prioritása?**

A UNIX rendszer folyamatai felváltva futnak, gyors váltásokkal. A prioritás-rendszer segítségével a rendszer hosszabb futásidőt oszt ki a fontosabb folyamatoknak.

**20. Mit tud az idézőjelekről a UNIX rendszerben?**

””: stringek deklarálása; különleges karaktereket (pl. változónevek) behelyettesíti.  
’’: minden karakter saját magát jelenti; nincs behelyettesítés.  
` `: parancs behelyettesítése.

**21. Mit jelent az stdin, stdout?**

stdin: standard input; alapértelmezett bemenet: billentyűzet  
stdout: standard output; alapértelmezett kimenet: monitor  
(stderr: standard error output; alapértelmezett hibakimenet: monitor)

**22. Hány szűrő kell egy csővezetékhez?**

Kettő, vagy több.

**23. Mondjon példát arra, hol használhat reguláris kifejezéseket?**

Csővezeték: pl grep parancs, sed parancs stb.

**24. Mi az az ASCII kódtábla?**

8 (eredetileg 7) bites karakterkódolási tábla, az angol ábécé kis- és nagybetűit, számokat, írásjeleket, és néhány vezérlőkaraktert tartalmaz.

**25. Mik azok a környezeti változók?**

Olyan globális változók, amelyek befolyásolhatják különböző folyamatok futását (pl. PATH, TEMP)

**26. Adja meg, hogy UNIX-ban milyen típusú(ak) lehet(nek) a változó tartalma(k)!**

UNIX-ban minden változó szöveges típusú.

**27. Mit jelent a parancsbehelyettesítés?**

Mikor egy parancs kimenetét egy szövegbe szeretnénk visszahelyettesíteni, akkor a szövegen belül, ` -k közé írjuk a parancsot. Futáskor lefut a parancs is, a végeredmény pedig bekerül a szövegbe (változóba stb.).

**28. Sorolja fel, hogy milyen műveletek (aritmetikai, logikai) léteznek UNIX shellben!**

Alapvetően csak szövegösszefűzés van közvetlenül a shellben. Más (pl. aritmetikai, logikai) műveletekhez parancsokra van szükség (pl. expr, bc, test).

**29. Melyik shell utasításnak van befejezési eredménye?**

Mindegyik shell utasításnak van befejezési eredménye: 0 = sikeresen lefutott, 1 = sikertelen.

**30. Hogyan implementálják a logikai értékeket a UNIX shellben?**

”Test” parancs segítségével (vagy ”[]”): lefut-e az adott parancs. Ezért 0 = igaz, 1 = hamis.

**31. Hogyan készíthetünk összetett logikai kifejezést UNIX shell scriptben?**

”Test” parancs –a vagy –o operátoraival egymás után fűzünk több kiértékelést. A precedenciát módosíthatjuk a ”\(|” és ”\)|” karakterekkel.

**32. Lehet-e paramétereket kezelő függvényeket definiálni UNIX alatt?**

Igen, C-szerű függvényeket lehet definiálni, amely a paramétereket shell script-hez hasonló módon kezeli; visszatérését a ”return” kulcsszóval lehet beállítani.

**33. Tudja-e (és ha igen, hogyan) futtatni a végrehajtási jogosultság nélküli shell scriptet?**

Igen, a script tartalmát át kell adni az adott shellnek, és azt futtatni. (sh -v, bár nem tudom, ez mit csinál)

**34. Mi az IFS?**

[Internal Field Separator] Alapértelmezett elválasztójel a shellben (alapesetben szóköz/tabulátor/soremelés); értékadással megváltoztatható (lehet több karakter).

**35. Milyen feladatokat tud elvégezni a SED-del?**

Komplex behelyettesítések, cserék (első, utolsó, összes elem cseréje; csere regex alapján stb.)

**36. Írja le általánosan egy SED parancs szintakszisát!**

sed [paraméter] [cím] s/minta/új\_minta/[jelző]  
([] = opcionális)

**37. Mi a különbség a SED használatában a " és a ' idézőjel használata között?**

": változók, különleges karakterek behelyettesítésre kerülnek.

': nem történik behelyettesítés.

**38. Jellemesse az AWK lehetőségeit!**

C nyelvi lehetőségeket ad; tipikus szűrő, pótolja a shell/sed hiányosságait; soronkénti szövegszerkesztést tesz lehetővé.

**39. Adj meg, hogy milyen parancsblokkok találhatók AWK-ban!**

3 parancsblokk:

Első sor előtti inicializálás: BEGIN { }

Soronkénti feldolgozás: { }

Utolsó utáni inicializálás: END { }

**40. Használható-e az AWK aritmetikai feladatok megoldására?**

Igen, C-ből ismert aritmetikai jelek működnek AWK-ban, ill. van numerikus függvénykészlete.

**41. Mi az MBR és mi a feladata?**

Master Boot Record (~Fő indítórekord): A merevlemez egyik particionált része, ahonnan az operációs rendszer betöltődik. A számítógép bekapcsolása után az alaplap itt keresi, és ide adja át a vezérlést.

**42. Írja le a UNIX-LINUX boot folyamatot!**

6 lépésből áll:

1. Bios

- a. Az alapvető be- és kimeneti folyamatokat biztosítja
- b. Megkeresi, betölti és lefuttatja a boot betöltő programot (MBR)

2. MBR

- a. Lásd: 41. kérdés

3. GRUB

- a. Grand Unified Bootloader ~ A főbb bootfolyamatokat ez indítja el
- b. Kialakítja a kezdeti RAM-ot
- c. Futtatja a Kernel-t

4. Kernel ~ rendszermag
    - a. Kiosztja a feladatoknak, hogy a hardvereket mennyire használhatják.
    - b. ~ multiplexálás
    - c. Átadja a vezérlést az init-nek
  5. Init
    - a. /etc/inittab néven van elmentve
    - b. Eldönti a Linux futásának a szintjét
    - c. 0: megáll, 6: reboot (nem érdemes)
  6. Futásszintű programok
    - a. Átadja a vezérlést a felhasználónak a további műveletekhez
- 43. Írjon le legalább egy UNIX-LINUX management lehetőséget!**  
File manager: Felhasználói felületet biztosít a felhasználónak a fájl – és mappakezeléshez a hierarchikus rendszerben és az alapvető fájlkezelő műveleteket is elvégzi (másolás, átnevezés std.)
- 44. Milyen hálózati kapcsolódási lehetőségeket ismer?**  
Fix: kábelalapú, Ideiglenes: WiFi, 3g, 4g, cellás, műholdas
- 45. Mit ért csomagkapcsolt hálózat alatt?**  
Nincs előre kiépített út a kommunikációra, továbbá a küldeni kívánt adatot csomagokra bontják, és ezek külön-külön kerülnek elküldésre.
- 46. Mit ír le az OSI modell?**  
[Open Systems Interconnection Reference Model] A számítógépek kommunikációjához szükséges hálózati protokollt határozza meg.
- 47. Nevezzen meg hálózati topológiákat!**  
Centralizált: Egy központi felületen (szervergépen, buszon) történik a kommunikáció.  
(csillag, gyűrű, teljes, fa)  
Decentralizált: Nincsen meg az a központi felület, amely a kommunikációt gyorsítaná, ezért a kiépítése vagy nem túl hatékony, vagy költséges (Sín, busz)  
Hibrid: előző kettő részenkénti alkalmazása (csillag-sín)
- 48. Mi a feladata egy switch-nek?**  
A hálózat felépítésében a kapcsolás és útválasztás feladatát, valamint a portok összekapcsolását a switch, mint hálózati eszköz valósítja meg.
- 49. Mi a feladata a routernek?**  
Hálózatok összekapcsolása és az ezeken történő adatfolyamatok irányítása, lebonyolítása.
- 50. Hogyan jellemzné az IPv4 címeket?**  
IP címek 32 bites szerinti felírása, bájtonként felírva egy 0 és 255 közti egész számmal, ponttal elválasztva.
- 51. Hol találkozik a DNS-sel az informatikában?**  
[Domain Name System] Számítógépes hálózatoknál, pl. internet
- 52. Mi az a DHCP?**  
[Dynamic Host Configuration Protocol] Dinamikus állomáskonfigurációs protokoll, ez osztja ki a lokális IP-t, hogy ne kelljen manuálisan konfigurálni.

**53. Milyen szerver elérési módozatokat ismer?**

SSH, TELNET, HTTP, FTP.

**54. Mire szolgál a HTTP protokoll?**

Állományok feltöltése, letöltése, főként HTML dokumentumokat.

**55. Mi történik, ha a public\_html könyvtárban nincs index.html fájl?**

Kilistázza a mappát és onnan lehet letölteni a tartalmát, vagy ha az privát, akkor hibát jelez a böngésző.

**56. Hogyan lehet jelszóval védeni egy weben lévő könyvtárat?**

.htaccess és .htpasswd fájllal.

**57. Mit ért virtuális host alatt?**

Egy webcímre más néven hivatkozunk.

**58. Mit jelent az SSI vagy CGI jog a webszerverekben?**

A webszerver dinamikus interfészeket (SSI, CGI) futtathat, az azokhoz szükséges paramétereiket (pl. stdio, környezeti változók) elérheti.

**59. Milyen Windows script lehetőségeket ismer? Van egyáltalán?**

Igen: Batch, WSH [Windows Scripting Host], PowerShell

**60. Mi biztosítja PowerShell-ben az „autoexec.bat” szerepét?**

gpedit.msc (ötletem sincs, hogy kéne értelmezni a kérdést, ennyit letem a diákok közt)

**61. Hogyan biztosítják PowerShell alatt a biztonságos script futtatását?**

ExecutionPolicy segítségével, amely alapértelmezetten “Restricted” - azaz nem engedélyez futtatást. Átállítás: Set-ExecutionPolicy –ExecutionPolicy Unrestricted. Lehetséges policy értékek: Allsigned, Remotesigned, Bypass.

**62. Milyen a PowerShell parancsok felépítése?**

Ige-[Modul]Főnév alakúak a parancsok. (pl: Get-Help)

Ige: minden műveletet hajt végre (pl: Get, Set, New, Remove)

Modul: megmondja milyen modulban található a parancs. Nem minden parancs tartalmaz modulnevet. Nem mindenhol szükséges.

Főnév: Milyen adaton vagy dolgon hajt végre valamit a parancs.

**63. Soroljon fel PowerShell-ben legalább két változóláthatósági formát!**

- global: mindenhol látható az egész PS-ben
- local: függvény vagy szűrő hatókörében lesz látható. Miután a függvény befejezte a futást a benne lévő változók elvesznek.
- script: a teljes scriptben látható lesz
- private: Olyan lokális változó amihez a „gyerek” környezetek nem férnek hozzá.

Egy környezetben definiált változókat, a környezetében használhatjuk, az ebből származó függvény, script látja. Azonos nevű esetén a lokálisat látjuk alapból. Változót definiálhatunk a scope-jával együtt: \$[scope:]név

**64. Hogyan irányítjuk át PowerShell-ben az output-ot? Lehet?**

Új fájl, felülírás: „Hello” > szoveg.txt

Hozzáfűzés: „szia” >> szoveg.txt

Nincs < vagy << fajta átirányítás!

**65. Hol és mire használható a dot sourcing?**

Függvény, változó szintjének a módosítása. Mivel függvényen belül is definiálható függvény és az nem hívható közvetlenül.

pl: ..\UtilityFunctions.ps1 futtatás után az UtilityFunctions script függvényei és változói használhatóvá válnak a jelenlegi scopeban is.

**66. Mit jelent a PowetShell függvények nevesített paraméterezi lehetősége?**

Van lehetőségünk elnevezni egy script paramétereit, és így nem az \$args változóba kerülnek.

```
param( $x, $y )
"A '$x={0}" -f $x
"A '$y={0}" -f $y
```

**67. Mi a különbség a mikroprocesszor és a mikrokontroller között?**

A mikrokontroller átalában egyetlen lapka ami tartalmaz egy processzor magot, adatmemóriát, programot és programozható ki/bemente perifériákat. Általában valamilyen konkrét feladatra optimalizált cél-számítógép.

A mikroprocesszor (CPU), néha kiegészítve memória vezérlővel. Bináris jeleket fogad és dolgoz fel. Más alkatrészekkel együtt szokták használni. Általános felhasználású.

**68. Mi a „Harvard architektúra” legfontosabb jellemzője?**

Egy számítógép-felépítési elv, amelyben a programkód és az adatok külön, fizikailag elkülönített útvonalakon közlekednek a processzor felé.

**69. Milyen operációs rendszerben lehet 128 bites egész számot definiálni?**

64 bites operációs rendszerben.

**70. Mire használható a lebegőpontos számábrázolás?**

Lehetővé teszi a valós számok tárolását és kezelését, véges tárhely esetében.

**71. Mit jelent az aszimmetrikus kódolás?**

Az aszimmetrikus kódolás azt jelenti, hogy az adatok kódolásához és dekódolásához két külön kulcsra van szükség egy helyett. Ez a két kulcs a publikus és a privát kulcs: a publikussal titkosítunk, és a priváttal lehet dekódolni (ehhez szükséges a publikus is).

**72. Hány számot takar az RSA publikus vagy privát kulcsa?**

Mind a kettő 2 számot takar (egy prímszámot – hatványkitevő a képletben, ez különbözik kulcsunként – és egy másikat, amire nézve azonos maradékosztállyba kerül a kódolt adat – ez azonos a privát és publikus kulcsnál)

**73. Mit jelent a bináris FTP lehetősége?**

Az üzenetküldő minden fájlt bájtonként küld el a vevőnek, a vevő a bájtfolyamot tárolja.

**74. Mit jelent a szöveges FTP? Létezik egyáltalán?**

Igen, szöveges fájlok küldésére alkalmas. Az adatot szükség esetén átalakítják 8 bites ASCII karakterkódra, a vevő fordított módszerrel dekódol.

**75. Hogyan irányíthatja át a szabványos bemenetet PowerShellben?**

Sehogy: nincs input átirányítás (<, <<) PowerShell-ben.

**76. Mi helyettesíti a "here input" funkciót PowerShellben?**

Read-Host –Prompt (nem vagyok biztos ebben, a „here input”-ról semmit nem találtam)

**77. Mire használható a profile.ps1 állomány? Van a UNIX-ban megfelelője?**

Globális PowerShell profil létrehozására, gyakran használt változókat, alias-okat és függvényeket lehet vele definiálni. UNIX megfelelője: \$HOME/.profile

**78. Mit értünk PowerShell modulon?**

Hasznos függvények, alias-ok, változók definíciójának gyűjtőhelyét.

**79. Elég-e a core PowerShell modul a registry módosításához? Miért?**

Igen, mivel a Core adatforrásai közé tartozik a Registry is.

**80. Hogyan használhatja PowerShellben a parancsbehelyettesítést?**

A „\$( parancs )” szintakszissal.

**81. Hogyan készíthet ciklust SED scriptben?**

Nincs ciklus SED scriptben

**82. Jellemzően milyen állományokat talál az /etc könyvtárban?**

Konfigurációs állományokat és a rendszer számára fontos adatbázisokat tartalmaz.

**83. Mire szolgál a hálózati csomagok TTL adata?**

A TTL [Time To Live] adat az adott csomag élettartama. Ha eléri a 0-t, a csomag törlődik. Olyan adatcsomagoknál hasznos, amelyek elkeverednek, nem jutnak el a vevőhöz.

**84. Mit mutat meg a "Netmask"?**

A LAN (alhálózat) méretét.

**85. Mit értünk "nem routolható" IP címen?**

Erre az IP-re a router nem tud adatot küldeni, pl. privát hálózatok.

**86. Mire szolgál a "gateway"?**

Ez az IP egy kivezető út, pl. /sbin/route -n.

**87. Mi az "ARPANET" és milyen lehetőségeket teremtett?**

[Advanced Research Projects Agency Network] A '60-as évek projektje. Csomagkapcsolt hálózat. Megteremtette: TCP/IP, FTP, Mail, NVP (Hangtovábbítás, kezdetleges).

**88. Mi az IPv6? Miért van rá szükségünk?**

128 bites (8 db 16 bites szám) címeket használó IP cím. Az egyre több internetre kapcsol készülék miatt az IPv4 modell előreláthatóan nem lesz elég, hogy az összeset megcímelje.

**89. Mondjon példát a "setuid" bit hasznosságára!**

Egy parancs a tulajdonosának jogosultságaival fussen.

**90 Mire jó a "sticky bit"?**

UNIX-ban könyvtárak jellemzője: könyvtárban csak saját fájl törölhető.

**91 Milyen célt szolgál az ACL használata Unix-Linux rendszerben?**

Hogy a hagyományos jogosultságrendszeren túl más jogokat is tudunk rendelkezésre bocsátani

**92 Létezik Windows rendszerben az ACL lehetősége?**

Igen.

**93 Mire használhatjuk a setfacl vagy getfacl parancsokat Linux rendszer alatt?**

Kiterjesztett jogosultságrendszerbeli (ACL) jogosultságokat állít be (setfacl) és ír ki (getfacl)

**94 minden fájlrendszerben hasonló módon (pl setfacl) lehet ACL jogokat állítani?**

Igen

**95 Mi az analóg- digitális jelek közti alapvető különbség?**

Az digitális jelek kvantáltak, időben és értékben diszkrétek (meghatározott értelkekkel vesznek fel). Az analóg jelek nem kvantáltak, tetszőleges értéket felvehetnek tetszőleges időben.

**96 Mi az adat, cím, vezérlő sín feladata?**

Címsín: eszközök címzését szolgálja, szélessége 32 vagy 64 bit.

Adatsín: a processzor adatokat küld vagy fogad ezen keresztül. 32 vagy 64 bit.

Vezérlő sín: a processzor vezérlőjeleket küld vagy fogad ezen keresztül. (min. 10-15)

**97 Hogyan készít szűrőt UNIX illetve Powershell alatt? Lehet?**

Igen, egy parancs után "|"-t írunk, majd ezt szűrőparancsok (grep, cut stb.) követnek.

**98 Mi a lényegi különbség a UNIX ls és a PowerShell Get-ChildItem parancsának eredménye között?**

ls string értéket ad vissza, Get-ChildItem objektumtömböt

ls mappákon és fájlokon dolgozik, Get-ChildItem képes pl. a registryben is navigálni

**99 Unix vagy Windows PS környezetben tud használni reguláris kifejezést?**

Unix shellekben a különböző parancsok támogatják, pl grep, sed

PowerShellben a string osztály támogatja a -split, -match és -replace metódusokban

**100 Milyen speciális jelentése van annak, ha Unix rendszerben egy fájlnév .-tal kezdődik?**

A fájl rejttetnek minősül, ls nem jeleníti alapból, ahogy a fájlkezelők sem.

**101 Mikor használhatóak jól a reguláris kifejezésekben létrehozható csoportok és miért?**

Amikor összetett keresésben pl. csak azt tudjuk, hogy 1 db betűt/számot/karaktert keresünk, de nem tudjuk, mi az.

x db karakter esetén: [a-z]{x}, ha betű, [a-zA-Z]{x}, ha kis- vagy nagybetű, [0-9]{x}, ha szám stb.

**102 Mi a lényegi különbség a UNIX shell és a PowerShell csővezetéken áthaladó adatok között?**

A UNIX shell csővezetékén szöveges adatok haladnak át, a PowerShell-én pedig objektumok.

**103 Milyen eszköztárral rendelkezünk UNIX és PowerShell szkriptek írásához?**

UNIX script: vi, vim, nano  
PowerShell: PowerShell ISE

**104 Milyen kiterjesztésűnek kell lennie egy PowerShell és egy shell szkriptnek?**

PowerShell: .ps, .ps1  
shell script: pl. .sh (nem kötelező kiterjesztés)

**105 Lehet paramétere egy szűrőnek? Ha igen, adjon meg egy tetszőleges példát, ha nem, miért nem!**

Igen, pl. cut -f1 -d";", vagy wc -w/-c stb.

**106 Mi a különbség az stdout, stderr csatorna között? Létezik PowerShellben?**

stdout: alapértelmezett kimenet; sikeresen lefutott programok ide írnak ki eredményt.  
stderr: alapértelmezett hibakimenet; hibaüzenetek ide kerülnek kiírásra.  
Igen, létezik PS-ben (Write-Output – Write-Error).

**107 Mi a "probléma" az egyes komplexensű számábrázolással?**

0-nak van egy pozitív és negatív alakja is: redundáns.

**108 Hogy lehet az stdin csatornát átirányítani Powershellben?**

PowerShell-ben nincs input átirányítás.

**109 Mire jó a SED? Mi a leggyakrabban használt parancsa?**

A SED szöveges adatfolyamban komplex behelyettesítésekre, cserékre jó. Leggyakoribb parancsa a 's/minta/új\_minta/'.

**110 Lehet-e egy SED scriptben shell scriptet hívni? Miért?**

Nem lehet, mert a SED csak behelyettesítéseket végez. (az indoklás saját interpretáció)

**111 Lehet-e shell scriptből SED scriptet hívni? Miért?**

Igen, lehet, mert a shell script tud másik scriptet futtatni, és sed parancsokat is végrehajt.

**112 Mi dönti el Unix rendszer alatt, hogy a script fájl milyen script?**

He egy shell szkriptek legelső sora a #! (shebang) karakter sorozattal kezdődik, a mögötte álló (teljes útvonalú) értelmező kapja a fájlt futtatásra.

```
#!/bin/sh
#!/usr/bin/bash
#!/usr/bin/python3
```

**113 Mit értünk az alatt, hogy egy processzor például 10 nanométeres technológiájú?**

A nanométer technológiájú nódus egykor a processzor kapuinak, tranzisztorainak méretére utalt.

Manapság marketing nevek a különböző gyártási technológiák között, és a név nem írja le az áramkör komponenseinek tényleges méretét

Kisebb MOSFET méretű processzorok energiahatékonyabbak, és potenciálisan gyorsabbak.

**114 Mit jelent, hogy egy processzor 32 vagy 64 bites?**

A memória címbusz szélességét írja le.

32 bites processzor 32 bit széles címeket tud lekérdezni, így ~3.5GB memóriát tud megcímzni, míg egy 64 bites ~16EB memóriát.

Minden modern processzor 64 bites, a mobilprocesszorok is.

**115 Hogyan tud AWK scriptet készíteni? Tud egyáltalán?**

Igen: #!/usr/bin/env awk -f

**116 Mit értünk IoT eszközökön, lehetőségeken? Lehet ezeket az eszközöket programozni?**

[Internet of Things] Internetkapcsolattal rendelkező mikrokontrollerek építhetők a minden nap használati tárgyakba, így azok távolról vezérelhetők, illetve különböző adatok lekérdezhetők róluk.

**117 Mi az AWK program BEGIN és END blokkjának a szerepe?**

Mind kettő egyszer fut le: a BEGIN az elején, az END a végén. Emiatt használható arra, hogy pl a BEGIN blokkban kiírjuk a feladat címét, azt elvégezzük, majd az END blokkban kiírjuk az összegzését.

**118 Lehet-e karakter kódtáblával adatokat, szöveget titkosítani?**

Lehet. Más kódtáblán más karaktert jelöl ugyan az a szám.

**119 Hogyan készíthet SED scriptben pl for ciklust?**

(:) -al létrehozott címekhez ugrik a vezérlés a b parancsal, feltétel teljesülésénél ugrik a t parancsal. Goto-hoz hasonló.

**120 Mi a különbség a PowerShell for ciklusa és Foreach-Object utasítása között?**

A ForEach-Object parancs végrehajt egy utasítást az adott objektum összes elemén. Az objektumot meg lehet adni csővezetéken keresztül vagy az InputObject paraméterrel.

Ez a dokumentum az ELTE IK Programtervező  
Informatikus hallgatói által készített, **nem hivatalos**  
segédanyag a Számítógépes Rendszerek c. tárgyhoz.  
Használat saját felelősségre!

## Készítők:

Ambrus-Dobai Márton, Bahrami Benedek Attila,  
Gortka Bence, Hadházi Dávid, Hamrik Szabin,  
Csépán Botond

2019 © ELTE IK

1. Milyen jellemző paraméterei vannak egy mai asztali számítógépnek?

A mai számítógépek túlnyomó többsége a Neumann-elvek alapján működik. Számítógépek fő részei: központi egység, operatív memória, perifériák, háttértárolók és a sínrendszer. A számítógép fő hardveregységei: alaplap, processzor, RAM, winchester, videókártya, tápegység.

2. Mi a cache szerepe a mikroprocesszorban?

A cache (gyorsítótár), a processzorra vagy a processzor környezetébe integrált memória, ami a viszonylag lassú rendszermemória-elérést hivatott felgyorsítani úgy, hogy előzetesen beolvassa azokat az adatokat és programrészeket, amelyekre a végrehajtásnak feltételezhetően szüksége lehet.

3. Milyen a kettes komplementű számábrázolás?

1 biten 2 különböző érték (0,1). Egész számok ábrázolása - jellemzően 4 byte-on tároljuk. Egy komplementű ábrázolás esetén az első bit lesz az előjel. 1 bájton így -127 +127 közötti számok ábrázolhatók. Kettes komplementű ábrázolás esetén egy nulla, -128 +127 közti számok ábrázolhatók egy bájton.

4. Mit tud az UTF kódolásról?

Az UTF-8 kódolás lényege, hogy a 7 bites ASCII kódtábla karaktereit (angol kis- és nagybetűk, számok és gyakoribb írásjelek) az ASCII-kódjukkal jelöli, az egyéb karakterek kódját pedig "feldarabolja" és a darabokat egy vezérlőjelet követő több, egymás utáni bájtból helyezi el úgy, hogy a bájtok mindegyike 127 felett van (azaz így nem téveszthető össze a 7 bites ASCII-kódok egyikével sem).

5. Soroljon fel legalább 3 memóriatípush!

A számítógép gyors elérésű és dinamikusan változó adattároló egységét hívjuk a számítógép memóriájának. Fontosabb memóriatípusok: ROM (csak olvasható memória), PROM (programozható ROM), EEPROM (elektronikuson törölhető PROM).

6. Soroljon fel olyan hétköznapi eszközöket, amelyekben "számítógép" található!

Mobiltelefon, sütő, gáztűzhely, mikro, stb..

7. Mi a különbség egy szerver és egy kliensgép között?

A kliens olyan számítógép amely hozzáfér egy (távoli) szolgáltatáshoz, amelyet egy számítógép hálózathoz tartozó másik gép nyújt. Jellemzői: kéréseket, lekérdezéseket küld a szervernek, a választ a szervertől fogadja, egyszerre általában csak kisszámú szerverhez kapcsolódik, közvetlenül kommunikál a felhaszbálóval.

A kiszolgávól vagy szerver olyan számítógépet, illetve szoftvert jelent, ami más gépek számára a rajta tárolt vagy előállított adatok felhasználását, a kiszolgáló hardver erőforrásainak (például nyomtató, processzor, háttértárrak) kihasználását, illetve más szolgáltatások elérését teszi lehetővé.

8. Soroljon fel legalább 3 operációs rendszert!

Microsoft - Windows XP, Windows 7, Windows 8

Apple/Macintosh - Mac OS X

...

9. Mi a különbség az ssh és a telnet kapcsolat között?

Az SSH szolgáltatás és protokoll egyben. A Telnethez hasonlóan távoli számítógépek elérésére és operációs rendszerük vezérlésére fejlesztették ki 1995-ben. A Telnet és az SSH között alapvető különbség, hogy az SSH biztonságos kommunikációs csatornát épít ki a kliens és a szerver között, mégpedig úgy, hogy nyilvános kulcsú titkosítást használ a kommunikáló gépek hitelesítésére, és az átvitt adatok bizalmasságának biztosítására.

10. Milyen szolgáltatásai vannak a mai operációs rendszernek?

- kliens - szerver különbségek
- közös, osztott háttértár használata
- közös nyomtatási szolgáltatás használata
- szervizek kezelése
- levelezés, web, terminál elérés stb.
- hálózati szolgáltatások (DNS, DHCP, stb.)
- felhasználók kezelése
- információs adatbázis

#### 11. Mit ért shell alatt? Nevezzen meg legalább kettőt!

Az operációs rendszernek kiadott parancsokat beolvasó és értelmező programokat az UNIX-terminológia parancsértelmezőknek, shellleknek (héjprogram, burok) nevezi. Ez a rendszerhéj az összekötő kapocs a rendszer magját képező kernel és a felhasználó között.

Unix alatt:

- Sh (Bourne shell)
- Ksh (Korn shell)
- Csh (C shell)
- Bash (Bourne again shell)

#### 12. Mi az az alias? Hol találkozott vele?

Ha egy gyakran használt parancsot rövidebb formában akarjuk elérni:

alias elnevezés = "parancs"

Shellscriptnél és powershellnél találkozhatunk vele.

#### 13. Milyen Unix fájlrendszer jellemzőket tud megemlíteni?

A Unix és a Unix-szerű operációs rendszerek minden eszközökhöz egy eszköznevet rendelnek, de ennek nincsen köze ahhoz, hogy hogyan is érhetők el az eszközön lévő fájlok. Valójában a Unix létrehoz egy virtuális fájlrendszert, amelyben minden eszközön lévő minden fájl egy hierarchiába rendeződik. Ez azt jelenti, hogy a Unix-ban van egy gyökérkönyvtár, és minden létező fájl valahol ebben a gyökértől induló struktúrában helyezkedik el. Sőt, ennek a gyökérnek nem is kell tényleges fizikai hely. Nem kell az első merevlemezen lennie, még csak a gépben sem kell lennie, a Unix képes egy megosztott hálózati erőforrást gyökérkönyvtárként kezelní.

#### 14. Milyen fájlrendszeret ismer?

Fájlrendszer típusok:

- lemezes fájlrendszer
- adatbázis-fájlrendszer
- tranzakcios-fájlrendszer
- hálózati-fájlrendszer
- speciális célú fájlrendszer

#### 15. Milyen fájlnév konvenciókat ismer Unix-ban?

-Név hossza nem korlátos

-Tetszőleges karakter használható

-Nincs kiterjesztés a Windows értelmezésében

-Ha a kezdőkarakter . (pont), akkor takart állományt hozunk létre

#### 16. Milyen fáj jellemzőket ismer Unix-ban?

-Név

-Méret

-Létrehozás dátuma

-Tulajdonos

- Tulajdonos csoportja
- Hard link szám
- Jogosítványok

17. Magyarázza el a Unix-ban levő alapvető hozzáférési jogosultsági rendszert!

Alapvetően egy 3x3-as rendszer él (oktális rendszer)

Minden bejegyzésnek van

-Tulajdonos jogosultsága (u)

-Csoport jogosultság (g)

-Mindenki más jogosultsága (o)

Minden jogosultság három részből áll

-R - olvasási jog

-W - írási jog

-X - végrehajtási jog

20. Mit tud az idézőjelekről a Unix rendszerben?

“ “: egyes karakterek vagy lefoglalt szavak speciális jellegét feloldják

‘ ‘: megakadályozzák hogy a \$-t a változók jelzésének tekintse

` `: operátor, mely a két ` között levő jesort végrehajtja és a kimenetét adja vissza úgy, hogy az új sor - karakter helyére szóközt rak

21. Mit jelent az stdin, stdout?

Standard input - a billentyűzet, alapértelmezett bemenet

Standard output - monitor, alapértelmezett kiírás

22. Hány szűrő kell egy csővezetékhez?

Kettő.

23. Mondjon példát arra, hol használhat reguláris kifejezéseket?

Szövegkereső, szöveghelyettesítő, szövegellenőrző feladatoknál alkalmazhatjuk őket.

24. Mi az ASCII kódtábla?

Szabványos ameriki kód. A kód jelkészlete az angol abc betűit, számokat, írásjeleket és vezérlő kódokat tartalmazza. Az ASCII jelkészlete 128 különböző szövegkaraktert a 0..127 előjel nélküli egész számokra képez le. ASCII-kódolást alkalmaznak a text-editorok (például a jegyzettömb) is.

25. Mik azok a környezeti változók?

A környezeti változók (angolul environment variables) a felhasználó bejelentkezésekor értéket kapnak, és a shellben bárhonnan elérhetőek.

26. Adja meg, hogy a Unix-ban milyen típusú(ak) lehet(nek) a változó tartalma(k)!

Lehetnek speciális változók, környezeti változók.

27. Mit jelent a parancsbehelyettesítés?

` : operátor, mely a két ` között levő jesort végrehajtja és a kimenetét adja vissza úgy, hogy az új sor - karakter helyére szóközt rak.

28. Sorolja fel, hogy milyen műveletek léteznek a Unix shellben?

-eq egyenlő

-ne nem egyenlő

-lt kevesebb mint

-le kevesebb mint vagy egyenlő

-gt nagyobb mint  
-ge nagyobb mint vagy egyenlő  
= két szöveg egyenlő  
!= két szöveg nem egyezik  
! a kijelentés eredményének negáltja

29. Melyik shell utasításnak van befejezési eredménye?

A sed parancs egy adatfolyamszerkesztő, aminek a segítségével egyszerű módosításokat végezhetünk a bemeneten érkező adatsorokon. Ha a sed szkriptben szereplő összes parancs (egy sorra több parancs is vonatkozhat) feldolgozta az aktuális feldolgozandó sort, akkor a soron végzett utasítások eredménye, az új sor tartalma kikerül a standard kimenetre.

30. Hogyan implementálják a logikai értékeket a Unix shellben?

Test parancsral a szkriptben logikai ellenőrzést hajtunk végre. Ha lefut a parancs, rendben elvégzi a feladatát, akkor logikai igaz értéket ad vissza. (0) Ha nem tudja rendesen elvégezni a feladatát, üres sort olvas be, hamis értéket ad vissza. (1)

31. Lehet-e paramétereket kezelő függvényeket definiálni a Unix alatt?

Igen.

32. Tudja-e futtatni a végrehajtási jogosultság nélküli shell szkriptet?

Nem.

33. Mi az az IFS?

Internal Field Separator, az alapértelmezett elválasztó helyett új elválasztó karakter definiálása

34. Milyen feladatokat tud elvégezni a sed-del?

A sed parancs egy adatfolyamszerkesztő, aminek a segítségével egyszerű módosításokat végezhetünk a bemeneten érkező adatsorokon.

35. Írja le általánosan egy sed parancs szintakszisát!

sed 'szkript' fájlnév1 fájlnév2 ...

36. Mi a különbség a sed használatában a " és a ' idézőjel használata között?

" használatakor a változó értéke helyettesítődik be, ' használatakor nincs érték behelyettesítés

37. Jellemzze az awk lehetőségeit!

alkalmazhatjuk szöveges fájlokon való műveletek végzésére. Míg a sed sorokon belüli tevékenységei korlátozottak, nincsen sok aritmetikai lehetőség valamint vezérlési szerkezetek is hiányoznak, addig az awk ezekre megoldást kínál.

éigolvassa a bemenetet, azt feldolgozza, majd a kimenetre ír.

az awk is soronként veszi a bemenetet (ezt a bemeneti sort gyakran aktuális rekordnak is nevezi), azt a kívánalmaknak megfelelően átalakítja, és a kimenetre írja ezt az eredményt.

Azt azutasítássort ami alapján a soronkénti átalakítást végezzük, az awk parancs programjánakis nevezük. Ezt a programot, ahogy a sed esetében is láttuk, fájlba is írhatjuk, és ezt afájt futtathatjuk.

38. Adja meg hogy milyen parancsblokkok találhatók az awk-ban!

BEGIN, END

39. Használható-e az awk aritmetikai feladatok megoldására?

Míg a sed sorokon belüli tevékenységei korlátozottak, nincsen sok aritmetikai lehetőség valamint vezérlési szerkezetek is hiányoznak, addig az awk ezekre megoldást kínál.

**40. Mi az MBR és mi a feladata?**

Személyi számítógépeken (PC) a rendszertöltés (boot) folyamatának részeként a BIOS betölti a merevlemez fő rendszertöltő rekordját (Master Boot Record, MBR), innen folytatódik az operációs rendszer betöltése. A fő rendszertöltő rekordban legfeljebb négy partíció adatainak tárolására van hely, ezért a merevlemez legfeljebb négy valódi partíciót tartalmazhat. Particionáláskor meg kell adni az aktív (boot) partíciót, ami MS-DOS és Windows rendszereken rendszerint az első elsődleges partíció, hogy a rendszer bootolásra képes legyen.

**42. Írjon le legalább egy UNIX-LINUX management lehetőséget!**

General purpose Linux computing

Hardware Support

Older Platforms

Backups & Restores

**43. Milyen hálózati kapcsolódási lehetőségeket ismer?**

vezetékes (LAN), vezeték nélküli, mobil szélessávú, VPN, DSL

**44. Mit ért csomagkapcsolt hálózat alatt?**

Két alapvető összetevőből állnak: a kapcsolóelemekből és az átviteli vonalakból. Ebben az esetben a tetszőleges hosszúságú üzenetek meghatározott terjedelmű csomagokban érkeznek meg, mivel a csomag hossza maximálva van. Amennyiben a csomagkapcsoló hálózatban a csomagmérést meghaladó üzenetet kell átvinni, akkor a forrásállomás az üzenetet részekre tördeli, és az egyes részeket egy egy csomag alakjában továbbítja. Az egyes üzenetdarabok így elszakadhatnak egymástól, és csak a célállomásnál áll össze belük a teljes egész eredeti üzenet. Egy csomagkapcsolásos hálózat egyidejűleg több üzenetet továbbít az átviteli vonalakon. Ezt az átviteli eljárást multiplexelés-nek nevezik. A csomagkapcsolás nagyon hatékonyan képes a vonalak kihasználására, mivel az adott két pont közötti összeköttetést több irányból érkező és továbbhaladó csomag is használja.

**45. Mit ír le az OSI modell?**

Az Open Systems Interconnection Reference Model, magyarul a Nyílt rendszerek összekapcsolásareferenciamodellje (OSI-modell vagy OSI-referenciamodell) egy rétegekbe szervezett rendszer absztrakt leírása, amely a számítógépek kommunikációjához szükséges hálózati protokollt határozza meg, s amelyet az Open Systems Interconnection javaslatban foglalt össze.

**46. Nevezzen meg hálózati topológiákat!**

busz, fa, lánc, gyűrű, csillag, horgolt

**47. Mi a feladata egy switch-nek?**

csillag topológiájú hálózatban manapság használt kommunikációs útvonalakat kiépítő csomópont. A hálózatba való első bekötésekor ugyanúgy viselkedik, mint a HUB, de „figyeli” a kiküldött csomagok „sorsát”. Az így begyűjtött információk alapján a 2 későbbiekben már célzottan, mindenkor megfelelő című géphez küldi a csomagot, így nem generál felesleges adatforgalmat.

**48. Mi a feladata a routernek?**

útvonalválasztó (külső/belső hálózat). Leggyakrabban a helyi hálózat internethöz való illesztésére használatos. Két címmel rendelkezik: a helyi hálózat félé látható belső címmel, melyen keresztül a LAN gépeivel kommunikál; illetve egy WAN-címmel, amelyen keresztül az internet többi szereplőjével kommunikál.

**49. Hogyan jellemezné az IPv4 címeket?**

IPv4 szerinti IP-címek 32 bites egész számok, amelyeket hagyományosan négy darab egy bájtos, azaz 0 és 255 közé eső, ponttal elválasztott decimális számmal írunk le a könnyebb olvashatóság kedvéért.

**50. Hol találkozik a DNS-sel az informatikában??**

A Domain Name System (DNS), azaz a tartománynévrendszer egy hierarchikus, nagymértékben elosztott elnevezési rendszer számítógépek, szolgáltatások, illetve az internetre vagy egy magánhálózatra kötött bármilyen erőforrás számára.

**51. Mi az a DHCP?**

A dinamikus állomáskonfiguráló protokoll (angolul Dynamic Host Configuration Protocol, rövidítve DHCP) egy számítógépes hálózati kommunikációs protokoll.

Ez a protokoll azt oldja meg, hogy a TCP/IP hálózatra csatlakozó hálózati végpontok (például számítógépek) automatikusan megkapják a hálózat használatához szükséges beállításokat. Ilyen szokott lenni például az IP-cím, hálózati maszk, alapértelmezett átjáró stb.

A DHCP szerver-kliens alapú protokoll, nagy vonalakban a kliensek által küldött DHCP kérésekből, és a szerver által adott DHCP válaszokból áll.

**52. Milyen szerver elérési módozatokat ismer?**

TCP, FTP

**53. Mire szolgál a HTTP protokoll?**

A HTTP (HyperText Transfer Protocol) egy információátviteli protokoll elosztott, kollaboratív, hipermédiás, információs rendszerekhez.

A HTTP egy kérés-válasz alapú protokoll kliens és szerver között. A HTTP-klienseket a „user agent” gyűjtőnévvel is szokták illetni. A user agent jellemzően, de nem feltétlenül webböngésző.

A HTTP általában a TCP/IP réteg felett helyezkedik el, de nem függ tőle. A HTTP implementálható más megbízható szállítási réteg felett is, akár az interneten, akár más hálózaton.

**54. Mi történik, ha a public\_html könyvtárban nincs index.html fájl?**

Minden webkönyvtár alapértelmezett fájlja az index.html, index.htm vagy index.php. Ez a gyakorlatban azt jelenti, hogy az oldalad látogatóit (<http://tedomainneved.hu>) a szerver valójában ide vezeti: <http://tedomainneved.hu/index.html>. Ez így működik az összes nyilvánosan elérhető könyvtárban, beleértve az aldomaineket is.  
ha nincs index.html fájl ez nem történik meg.

**55. Hogyan lehet jelszóval védeni egy weben lévő könyvtárat?**

Néha szükségünk lehet arra, hogy a weboldalunkra feltöltött bizonyos mappát/mappákat jelszóval levédjünk. Ennek több megoldási lehetősége van, de a htaccess a legegyszerűbb. Ezt Apache szerver esetén bárki megteheti (legtöbb tárhely szolgáltatónál ez van).

**56. Mit ért virtuális host alatt?**

Képessé tesz egyetlen álló gépet arra, hogy töbszörös domainnal rendelkező web server legyen.

**57. Mit jelent az SSI vagy CGI jog a webszervereken?**

SSI: Szerver oldali beágyazás- egy olyan programra való hivatkozás, melyet a szerver a kért weboldal elküldése előtt lefuttat és eredményét a html dokumentumba küldés előtt beépíti.

**58. Milyen Windows szkript lehetőségeket ismer? Van egyáltalán?**

Python, c#, php,xml, nullscript, pascal, java

**59. Mi biztosítja PowerShell-ben az "autoexec.bat" szerepét?**

Az AUTOEXEC.BAT egy olyan fájl, ami a gép indulásakor végrehajtandó parancsokat tartalmazza. Ennek szerepét a PowerShellben az ún. profilok veszik át. A profilok egyszerű fájlban tárolt szkriptek, amelyek a PowerShell indításakor automatikusan lefutnak.

60. Hogyan biztosítják PowerShell alatt a biztonságos szkript futtatását?

A providereket úgy kell elképzelni, mint „szolgáltatókat”, amelyek egyfajta „távvezérlést” tesznek lehetővé bizonyos eszközökhöz. Ezek az eszközök lehetnek lokálisak, vagy távoli gépen futtatott szolgáltatások. Tehát nem közvetlenül az adott eszközöt vesszük igénybe, hanem a provider utasítjuk egy-egy utasítás végrehajtására. Azt, hogy ezt miképp hajtja végre, már nem a mi problémánk.

61. Milyen a PowerShell parancsok felépítése?

Cmdlet \_\_ A paraméter neve \_\_ A paraméter értéke

62. Soroljon fel PowerShellben legalább két változó láthatósági formát?

A PowerShell-ben 4 láthatósági szintet különböztetünk meg:

Global: a globális hatókör az egész PowerShell-re érvényes. Bármelyik munkamenetből elérhetők.

Local: Egy adott függvény vagy szűrő hatáskörébe eső változók láthatóságát nevezzük lokálisnak.

Miután a helyi hatókör befejezte futását, a benne lévő változók elvesznek.

Script: Egy adott munkamenetben érvényes változók. A munkamenet befejeződésével a változók megsemmisülnek.

Private: A zárt hatókör megegyezik a lokális láthatósággal, azzal a különbséggel, hogy a gyermek hatókörökbe nem öröklődnek a belső hatókör tartalmába eső privát változók.

63. Hogyan irányítjuk át PowerShellben az outputot? Lehet?

Igen lehet pl.:

I)"Hajrá Fradi!">fardi.txt #felülírás, új fájl

II) Get-Content fradi.txt #PS típus

Cat fradi.txt #unix típus

Type fradi.txt #dos típus

III)"Hajrá UTE!">>fradi.txt #hozzáfűzés, ha nincs fájl, létrehozza

Itt nincs < vagy << átirányítás

64. Hol és mire használható a dot sourcing?

Amikor dot source-ingolsz egy szkriptet, az összes változó és funkció meg lesz határozva a szkriptben, és addig fennál a shellben, ameddig a szkript véget nem ér.

65. Mit jelent a PowerShell függvények nevesített paraméterezi lehetősége?

(Kiindulva abból, hogy a PS valójában C#Szkript) Nem szükséges betartanod azt, hogy egy függvénynek milyen paraméterei és milyen sorrendben következnek. Tehát ha például ha van egy a(b, c) függvényed, akkor te meghívhatod fordított sorrendben is, előbb a c, majd a b paraméterét megadva, persze ki kell mondani, hogy mi mit jelent: a(c: 5, b: 9). Ekkor a megfelelő változónak a megfelelő érték lesz adva.

66. Mi a különbség a mikroprocesszor és mikrokontroller között?

A mikrokontroller egy mikroprocesszor kiegészítve az áramköri lapkájára integrált perifériákkal. Régebben mikroprocesszor-típusokat használtak a vezérlési feladatok elvégzésére. A mikroprocesszor használatakor a szükséges perifériák miatt további IC-keket kellett beépíteni. Az áramköri technológia fejlődésével egyre több perifériát az IC tokba lehetett integrálni, így alakult ki a mikrokontroller, nagyon tömör áramkört eredményezve.

67. Mi a "Harvard architektúra" legfontosabb jellemzője?

A Harvard-architektúra egy számítógép-felépítési elv, amelyben a programkód és az adatok külön, fizikailag elkülönített útvonalakon közlekednek a processzor felé.

**68. Mire használható a lebegőpontos számábrázolás?**

A számítástechnikában a lebegőpontos számábrázolás lehetővé teszi a valós számok kezelését véges tárhely esetében, széles skálát fedvén le a számhalmazon belül. A számok rögzített számú számjegyekkel ábrázolhatóak, és egy kitevő (exponens) segítségével vannak skálázva. A skálázás alapja leggyakrabban 2, 10 vagy 16.

**69. Mit jelent az asszimetrikus kódolás?**

Ennél a kódolási eljárásnál két, egymást minden kiegészítő kulcs létezik. Az egyik kulcs - a nyilvános kulcs (Public Key) - az üzenet kódolására, a másik - a titkos kulcs (Private Key) - a dekódolásra. A két kulcs együtt egy kulcspárt alkot.

A dologban az a különös, hogy az egyik kulcsból a hozzá tartozó másikat egyáltalán nem könnyű kitalálni vagy kiszámítani. Ezért a kulcspár egyik tagját nyilvánosságra hozhatjuk.(PL:két szám összeszorzás)

**70. Mit jelent a bináris ftp lehetősége?**

A fájlátviteli protokoll (FTP: file transfer protocol) a gépek közötti hálózati fájlátvitel segédeszköze. Lehetővé teszi a fájlok mozgatását a kliens és szerver között minden irányban, könyvtárak létrehozását, átnevezését, illetve törlését.

Bináris FTP-kapcsolat esetén az átküldött fájlok bájthelyesen, változtatás nélkül érkeznek meg.

**71. Mit jelent a szöveges ftp? Létezik egyáltalán?**

Szöveges kapcsolat esetén az FTP gondoskodik az eltérő szöveges fájlformátumok átalakításáról (például a DOS–UNIX szövegfájlok konverziójáról). A helyes átviteli mód kiválasztása az átvitel előtt a felhasználó feladata. Alapértelmezésben az FTP a szöveges módot használja, ami sok hibára ad lehetőséget.

**72. Hogyan irányíthatja át a szabványos bemenetet PowerShellben?**

pipeline annyit jelent, hogy egy adott commandlet kimenete (tehát a parancs után tesszük a | vonalat, ami magyar kiosztású billentyűzeten az AltGr-W kombinációra hozható elő) lesz a következő parancs bemenete (a következő parancs az előző parancs által kiadott adatokkal fog dolgozni).  
Get-ChildItem | Out-File out.txt

**74. Mire használható a profile.ps1 állomány? Van a UNIX-ban megfelelője?**

Minden indításkor autómatikusan lefuttathatóak parancsok, scriptek stb. Az UNIX megfelelője a .profile állomány

**75. Mit értünk PowerShell modulon?**

A PowerShell a moduláris felépítésének köszönhetően nagyon sok funkcióval bővíthető. Ezeket a funkciókat modulok vagy providerek formájában tudjuk használni a legegyszerűbben. Úgy kell ezt elkapozni, hogy például van egy új programunk ,ami valamilyen szolgáltatást valósít meg. Ekkor lehetőségünk van egy olyan modult írni hozzá, amelyen keresztül képesek vagyunk kezelní (mi vagy épp mások) a PowerShell alapvető parancsainak a segítségével. Például új felhasználókat tudunk felvenni, adatbázisokat manipulálni, vagy csak egyszerűen egy távoli gép beállításait kezelní, frissíteni.

**76. Elég-e a core PowerShell modul a registry módosításához? Miért?**

Nem, mert core feladata: Szokásos grafikus lehetőségek (File, Edit, View) • Tools – munka ablak font, szín beállítások. • Debug – Szokásos nyomkövetési lehetőség! Kiegészítés szükséges.

77. Hogyan használhatja PowerShellben a parancsbehelyettesítést?

- parancs` parancsot végrehajtja, parancs kimenete kerül a helyére
- Bash shellben: \$(parancs) a='date' ; b='date' # • echo \$a # ??? A date szó lesz az eredmény.

78. Hogyan készíthet ciklust sed scriptben?

```
cat osztaly|sed 's/3/9/g ; s/9/21/' # Cseréljük az osztály soraiban az összes 3-at 9-re, majd az első kilencet 21-re minden sorban!
```

79. Jellemzően milyen állományokat talál az /etc könyvtárban?

A FHS alapján a /etc mappába az aktuális gép szintjén (rendszer szinten) érvényes, statikus konfigurációs állományok (hosztnév, ifconfig, dns resolution, webszerver hosztok, ssh security) kerülnek.

80. Mire szolgál a hálózati csomagok TTL adata?

A Time To Live (TTL) számítógép-hálózatok működésével kapcsolatos fogalom. Szó szerinti jelentése: élettartam. A TTL mezőre azért van szükség, mert nélküle redundáns hálózatoknál hibás beállítás esetén előfordulhatna, hogy az IP csomagok körbe-körbe keringenek. A TTL mező nélkül a hálózatban a végtelenségig keringő csomagok jöhetnének létre és használhatatlanná válna mivel folyamatosan foglalnák a hálózat sávszélességét.

80. Mire szolgál a hálózati csomagok TTL adata?

A TTL mezőre azért van szükség, mert nélküle redundáns hálózatoknál hibás beállítás esetén előfordulhatna, hogy az IP csomagok körbe-körbe keringenek. A TTL mező nélkül a hálózatban a végtelenségig keringő csomagok jöhetnének létre és használhatatlanná válna mivel folyamatosan foglalnák a hálózat sávszélességét.

81. Mit mutat meg a "Netmask"?

"hálózati maszk"-ra. (netmask) A netmask ugynúgy néz ki mint egy IP cím, tehát 4 pontokkal elválasztott decimális szám. A hálózati maszk azonban azt mutatja meg, hogy egy adott alhálózaton, az elejétől fogva, hány bitnek kell megegyeznie az IP címben. Ha például az első 24 bitnek kell megegyeznie, akkor az alhálózati maszk a következő lesz:

Mask: 255.255.255.0 – megadja a LAN méretét (256)

Netmaszk

82. Mit értünk "nem routolható" IP címen?

az előtag mező értéke a bináris 1111111010. Ezt 54 nulla követi, aminek következtében a kapcsolati szintű címeknél az összes hálózati előtag ugyanaz, ezért nem routolhatóak.

83. Mire szolgál a "gateway"?

- Gateway IP: a kivezető út IP címe (router)

84. Mi az "ARPANET" és milyen lehetőségeket teremtett?

A hálózat űsprojektje: ARPANET. NCP a kezdeti ARPANET kommunikációs szabvány.  
ARPANET szolgáltatások • File transfer (FTP – RFC354, 1971,73) • Terminál szolgáltatás (telnet – RFC 137, 1971, RFC854) • Erőforrások megosztása (NFS) • Üzenetek továbbítása (Mail- RFC524,561 1971,73) • Hang továbbítás (NVP) – nem sikeres, ma helyette: VOIP!

85. Mi az IPv6? Miért van rá szükségünk?

- Születésének fő oka: IPv4 előrelátható szűkössége!
- IPv6 128 bites címeket használ!
- 8 darab 16 bites szám hexa alakja:
- 2015:0a0d:0102:1961:0324:fe01:03ab:0405
- Első 64 bit: subnet prefix

Második 64 bit: interfész azonosító

- Első standard: RFC2640 (1998 dec.)

86. Mondjon példát a "setuid" bit hasznosságára!

Futtatható fájl minden tulajdonosa, illetve csoportja jogosultságaival fusson. Ez a gyakorlatban az effektív uid/gid értékeket állítja, a program a setuid/setgid rendszerhívásokkal visszatérhet a valós uid/gid használatára.

87. Milyen célt szolgál az ACL használata Unix-Linux rendszerben?

- Hozzáférés-szabályozási listák (Access Control List, ACL)
- ( minden állományhoz egyenként adhatunk felhasználókat különböző jogokkal!)

88. Létezik Windows rendszerben az ACL lehetősége?

Nincs.

89. Mi az analóg- digitális jelek közti alapvető különbség?

Analóg jel, információ, folytonos jelértékek! A környezeti paraméterek, távolság, hőmérséklet, zene, zaj, áramerősség stb. természetes értékei! • Digitális jel, információ, diszkrét, nem folytonos értékek tárolása!

90. Mi az adat, cím, vezérlő sín feladata?

vezérlő sín: lehetővé teszi adatok vagy tápfeszültségek továbbítását a számítógépen belül vagy számítógépek, illetve a számítógép és a perifériák között.

adat: Az adat elemi ismeret. Az adatokból gondolkodás vagy gépi feldolgozás útján információkat, azaz új ismereteket nyerünk.

cím: egyértelműen azonosítja

91. Hogyan készít szűrőt UNIX illetve Powershell alatt? Lehet?

Fontosabb kész szűrők: • cut, tee, sort, uniq, wc, grep

Szűrő formában |

Szűrő példa: • cat nevsor|grep Pista # Eredményül kapjuk a Pista-t tartalmazó sorokat.

92. Mi a lényegi különbség a UNIX ls és a PowerShell Get-ChildItem parancsának eredménye között?

Mind2 az aktuális könyvtár tartalmát listázza ki ,csak az ls nem írja ki a jogosultságokat es a utolsó módosítás dátumát.

93. Melyik környezetben tud használni reguláris kifejezést a UNIX, a Windows PowerShell világában vagy mindkettőben:

Mindkettőben.

94. Milyen speciális jelentése van annak, ha egy fájlnév .-tal kezdődik?

Ha a kezdőkarakter .(pont), akkor takart állományt hozunk létre.

95. Mikor használhatók jól a reguláris kifejezésekben létrehozható csoportok és miért?

? Egy zárójelbe () tett reguláris kifejezés. Az a szöveg felel meg neki, ami a zárójelben szerepel. Egyfajta csoportosító szerepe van. Például ezekre a csoportokra a sed programban lehet a sorszámkossal hivatkozni. Az ?(alma|dió)fa? reguláris kifejezésnek az ?almafa? és a ?diófa? szöveg felel meg.

96. Mi a lényegi különbség a UNIX shell és a PowerShell csővezetéken áthaladó adatok között?

Powershell : több parancsot kell egymás után kiadnunk, és ezeknek a parancsoknak az eredményével szeretnénk tovább dolgozni.

Unix: egyik program kimenetét a másik program bemenetével köti össze.

. Unix shell text alapú, tehát ha pipeolsz akkor csak a szöveg megy át. Powershell objektum alapú, pipenal mindenfele tulajdonsagot külön tudsz lekerdezni az objektumból, ergo tudsz groupolni is meg barmit smile hangulatjel

98. Milyen kiterjesztésűnek kell lennie egy PowerShell és egy shell szkriptnek? Van egyáltalán előírás vagy szabadon megválasztható?

PowerShell szkriptek .ps1 (bár nem vagyok benne biztos, hogy ki van kényszerítve... megfelelő társítás esetén lehet, hogy a .fradi is fut!), Linux esetén -úgy mint semmi más fájl esetén sem Linux alatt- semmi nincs kikényszerítve... általában nem is használják, de ha nagyon muszáj, .sh-nak szokás elnevezni. Linuxnak a shebang line kell a fájl elejére, hogy a shell tudja, hogy neki most shell (és hogy milyen shell) szkriptet kéne futtatnia.

99. Lehet paramétere egy szűrőnek? Ha igen, adjon meg egy tetszőleges példát, ha nem, magyarázza meg, hogy miért nem lehetséges!

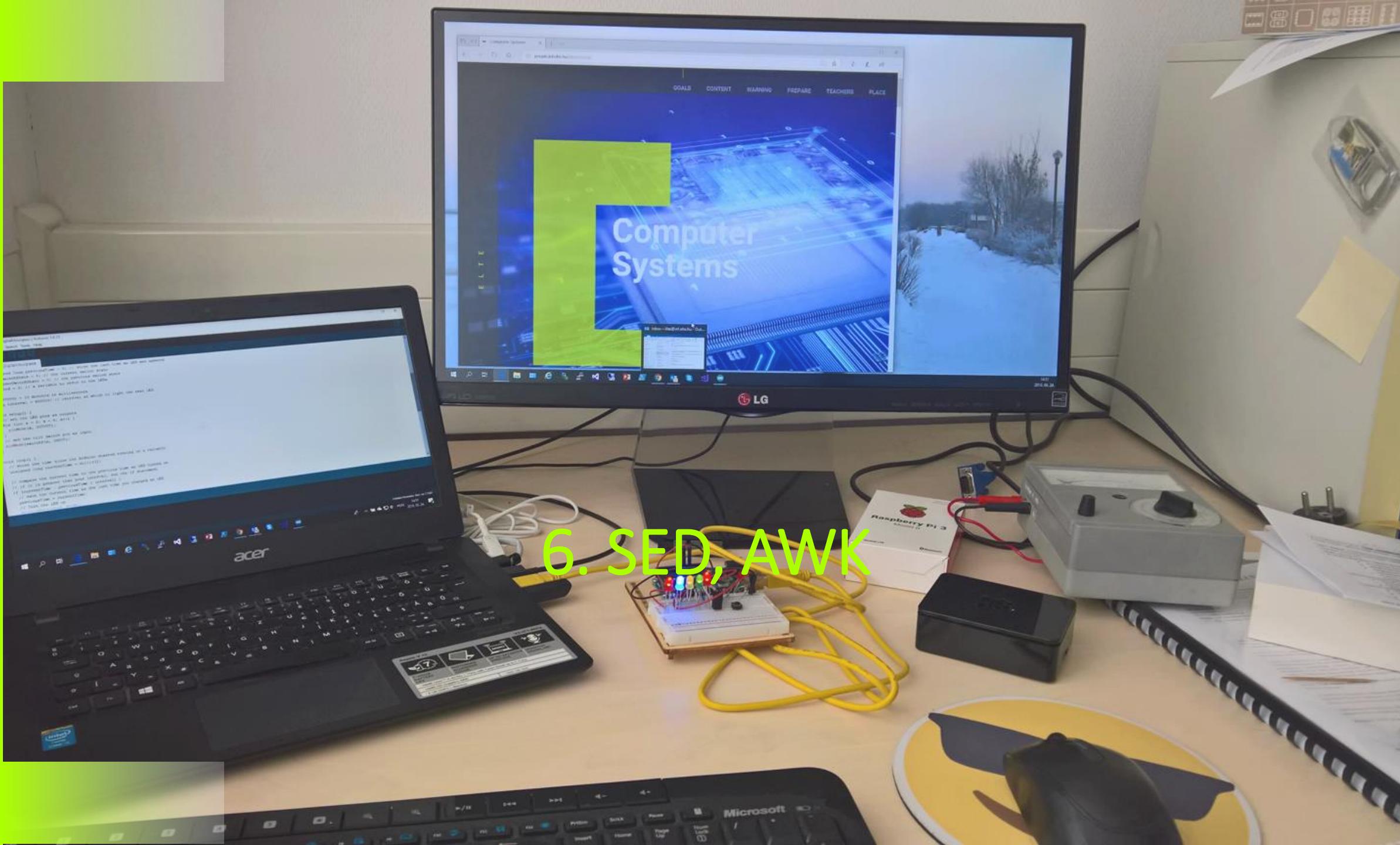
Igen, cat nevsor | grep Pista # Eredményül kapjuk a Pista-t tartalmazó sorokat

100. Mi a különbség az stdout, stderr csatorna között? Léteznek PowerShellben?

stdout (1) - monitor, alapértelmezett kimenet

• stderr (2) – monitor, alapértelmezett hibakimenet

Nemes egyszerűséggel annyi, hogy két külön fájlcsoport. Az stdout-ra a szabványos kimenetek (tehát minden olyan érdemi kimenet és felhasználói interakció) mennek, az stderr-re pedig a hibák. Ez két külön stream, külön módon kaphatóak el és értékelhetők ki. (Lásd: progalap és bíró... a Bíró csak az stdout-on kapott kimenetet fogja el, rágja meg és próbálja lenyelni, az stderr-t figyelmen kívül hagyja.) Unix világban fontos a megkülönböztetés, mert a legtöbb parancssori eszköz kimenetét valami módon tovább pipeoljuk egy másik feldolgozóba... és ha mondjuk a kimenet közé bekerül egy nem várt hibaüzenet, akkor a későbbi feldolgozás problematikussá válhatott. Szokásos, hogy a rendes kimenetek amiket az elkapó programnak kezelnie kell az megy tovább a pipe-on, a hibaüzenetek meg szorgosan jönnek a terminálra.



# Visszatekintés

- Számítógépek, információk, számábrázolás, kódolás
- Felépítés, kliens-szerver szerep,fájlrendszerek
- Alapvető parancsok, folyamatok előtérben, háttérben
- I/O átirányítás, szűrők,reguláris kifejezések
- Változó, parancs behelyettesítés,aritmetikai, logikai kifejezések
- Vezérlési szerkezetek

# Mi jön ma?

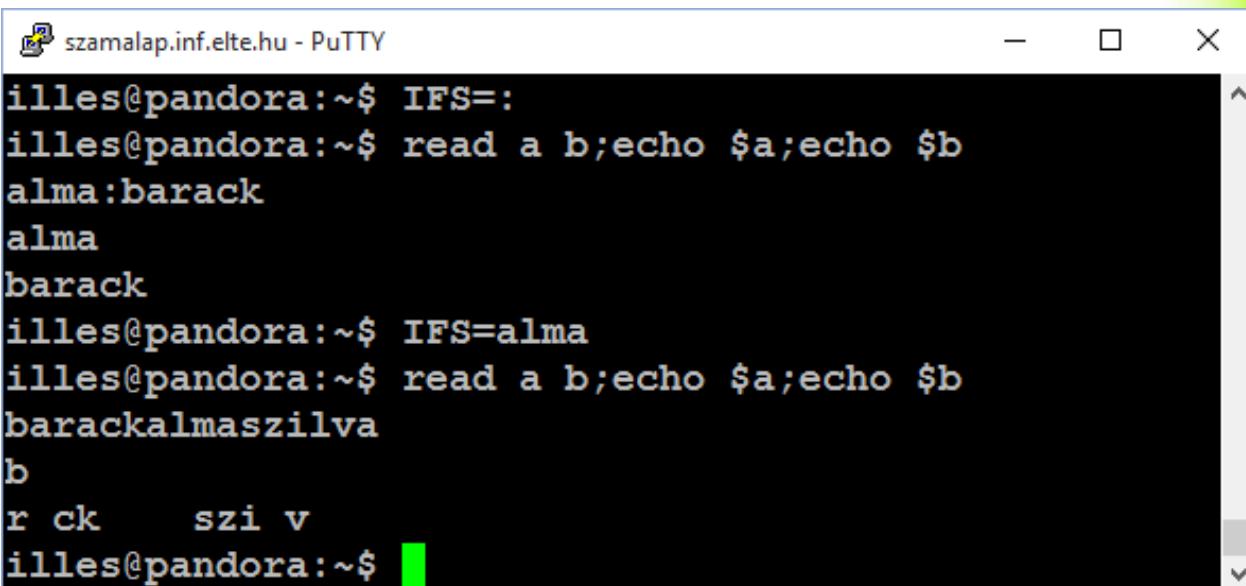
- Még több szűrés!
  - SED
- Még több programozási lehetőség!
  - AWK

# Input –Output utasítások

- Legtöbb parancs szabvány kimenetre ír
  - echo parancs, -n nincs új sor
- Bemenetről olvas:
  - read parancs
    - Példa: read alma #alma változóba olvas enterig
    - read korte; echo \$korte # ok
    - echo szia|read a; echo \$a # üres, a read nem szűrő
  - line parancs - szűrő
    - Standard inputról olvas egy teljes sort, amit a standard outputra ír
    - echo szia|`line`
    - echo alma barack|for i in `line` ; do echo \$i; done # eredmény: alma, barack külön sorban

# IFS - Internal Field Separator

- Több Unix implementációnak része!
  - BASH tartalmazza.
- Az alapértelmezett elválasztó helyett(helyköz, tab) új elválasztó karakter definiálása!
  - Célszerű egy karaktert megadni!
  - IFS=: #kettőspont az új elválasztó



```
szamalap.inf.elte.hu - PuTTY
illeg@pandora:~$ IFS=:
illeg@pandora:~$ read a b;echo $a;echo $b
alma:barack
alma
barack
illeg@pandora:~$ IFS=alma
illeg@pandora:~$ read a b;echo $a;echo $b
barackalmaszilva
b
r ck      szi v
illeg@pandora:~$
```

# Szűrjünk még...

- Mit tudunk eddig szűrni?
  - Sorból kivágni karakter, mező elemeket (cut)
  - Teljes sorokat (grep, reguláris kifejezések)
- Elég ez?
- Mit nem tudunk?
  - Sok minden....például nem tudunk soron belül szövegrészeket keresni, cserélni!

# SED – Stream EDitor

- Szűrő – a bemeneti sorokon a megadott szerkesztési műveleteket, módosításokat végzi (edit)
- Feladata: Komplex behelyettesítések, cserék a szabványos bemenetre érkező sorokon, eredmény a szabványos kimeneten. Veszi az összes sort (ciklus), majd minden egyes soron a parancsai végrehajtódnak, a módosított sor a kimenetre kerül!
- Példa: cat osztaly | sed 's/3/9/g' # A "s/3/9/" idézőjel is jó!
  - Keressük meg a sorokban az összes (g) 3-t, majd ezt cseréljük ki 9-re.

# sed fontosabb paraméterek

- -n, Nincs automatikus kiírás a szabvány kimenetre. Csak a sed script kiíró utasításai írnak a kimenetre.
- -f scriptfile, A paraméterül megadott fájlban van a program, a sed script. Jellemzően egy sorba egy parancsot írunk, de a ; elválasztóval több parancs is kerülhet egy sorba.
- -e A sed parancssorában több script van. Egy esetén elhagyható.

# sed script utasítások

- Több parancs megadása közvetlen script parancsban:
  - ; Pontosvessző a parancsok között!
    - Példa: cat osztaly|sed 's/3/9/g ; s/9/21/' # Cseréljük az osztály soraiban az összes 3-at 9-re, majd az első kilencet 21-re minden sorban!
    - Külön sorba írhatók a parancsok, másodlagos promptot kapunk!

```
$ cat osztaly | sed 's/3/9/g  
>s/9/21/' [enter]  
zoli 21 4 2 5 4  
feri 2 21 4 5 9  
juli 4 21 2 9 9  
$
```

# Sed script készítés

- Parancssorban több parancs elhelyezése lehetséges, de nem szerencsés!
- Készítsünk scriptet!
  - chmod +x sedpelda
  - Futtatás: sedpelda osztaly

```
teszt@pandora:~/> cat sedpelda
#!/bin/sed -f
#
s/3/9/g #3-9 csere mindegyik sorban, összes 3-es csere
s/9/10/ #9-10 csere mindegyik sorban, csak első 9-es csere
```

# sed fontosabb parancsok I.

- Sed parancs alakja: sed [par] [cím] s /minta/új\_minta/[jelző]
  - Feladata: Keressük mintát, cseréljük új mintára!
  - A [par] a sed paraméterei, a fontosabbakat láttuk, többi: man
- Jellemzően a minta reguláris kifejezés, ezt keressük.
- A cím, amire vonatkozik a sed parancs, ez lehet reguláris kifejezés is, jellemzően szám, vagy intervallum 1,10 s/.../.
- \$ jel az utolsó sort jelenti # 5,\$s/alma/körte/
- ! jel, a tagadás operátora
  - 2!s/alma/körte/ # a második sor kivételével
  - # minden sorban csere

# Egyéb cím meghatározás

- Ha nincs megadva, akkor minden sorra vonatkozik s parancs!
- N – az N. sorra vonatkozik, Pl: 4/3/9/g # a 4. sorban cserélünk
- $x \sim y$  – x. sor majd utána minden y. sor!
  - Példa: 3~2/3/9/g # a 3. sortól kezdve minden második sor!
- $x+y$  – x. sor majd utána a következő y sor!
  - Példa: 3+2/3/9/g # a 3. sortól kezdve két sor
- A teljes cím megadáshoz lásd referencia!

# A keresés-csere parancs jelző értékei

- A jelző értékei:
  - n: sorszám, a cserét az n. mintán kell elvégezni, ha elmarad, n=1. Ha n nagyobb mint utolsó előfordulás, a csere nem csinál semmit!
  - g: Az összes mintát le kell cserélni.
  - p: Kilistázza az aktuális sort! (pg együtt is lehet)
  - w fájl: Menti fájlba(hozzáfűz) az aktuális sort!
  - r fájl: Beolvassa a bemenetre a fájl tartalmát!

# sed fontosabb parancsok II.

- /új\_minta &/ Új mintát a minta elejére teszi
  - Pl: echo fradi|sed 's/fradi/hajra &/' #hajra fradi
  - A & jel jelenti a régi mintát, bárhol szerepelhet!
- \szám használata,
  - \1 az 1. reguláris kifejezés,
  - \ elnyomja a következő metakarater hatását(\.ali)
  - \n , soremelés beszúrása (\n >\2/'), a példában nincs n mert közvetlenül parancsként írjuk be!

```
illes@panda:~$ echo .ali 4 Ali baba|sed 's\.\ali \([0-9][0-9]*\) \(.*)\1. rész\> \2/' (enter)
4. rész
Ali baba
illes@panda:~$
```

# sed fontosabb parancsok III.

- Törlés: [címtartomány]d
  - d parancs törli a címtartomány sorait
  - Pl: cat osztaly | sed '1d; s/3/9/pg; s/9/21/' #első sort törli, majd a többi soron a másik két parancs
- Hozzáfűzés: a
  - Pl: cat osztaly | sed 'a\alma' # új sorként az alma, minden sor után
- Beszúrás: i
  - Pl: cat osztaly | sed '2,3i\alma' # 2,3 sor elő Alma

# sed fontosabb parancsok IV.

- y – minta karakter csere: echo papagáj|sed 'y/ag/uh/' #pupuháj
  - A cserélendő karakterszám azonos a cserélt karakterek számával!
- q – kilépés, adott címsor után a sed kilép
- :címke – címke készítés
- b címke – feltétel nélküli ugrás
- t címke – feltételes ugrás, ha volt sikeres csere

# Sed példa I.

- 1. példa:

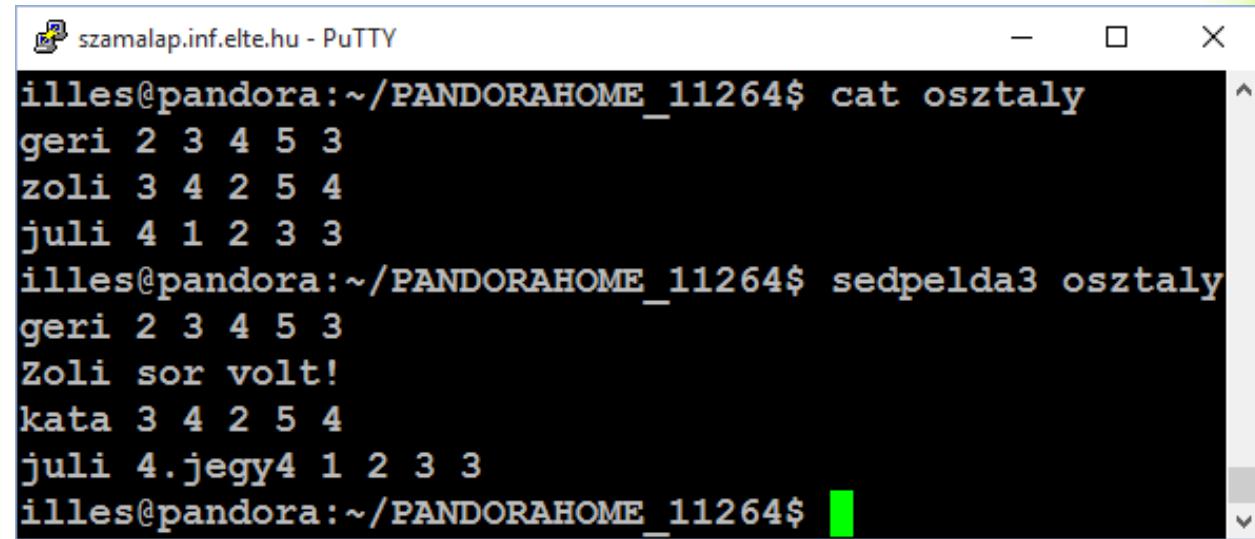
```
#!/bin/sed -f
#
# Az első és második sorban számok mögé írunk
1,2s/\([1-9][1-9]*\)/.&.kívánság\1/g # minden talált szám után!!!
# A második sor után quit!
2q
```

# SED program idézőjelek

- A SED paraméterként megadott programot idézőjelek között kell megadni!
- Ez lehet az egyszeres ', és a dupla " is!
- Van különbség!
- `x=Tibi; echo Laci ügyes! | sed "s/Laci/$x/"`
  - Eredmény: Tibi ügyes!
- `x=Tibi; echo Laci ügyes! | sed 's/Laci/$x/'`
  - Eredmény: \$x ügyes!

# Sed példa II.

```
#!/bin/sed -f
#
s/zoli/kata/ # zoli csere katára
t zolisor # feltételes ugrás, ha volt sikeres csere
# Mindegyik sorban a számok mögé írunk
2,$s/\([1-9][1-9]*\)/&.jegy\1/ #csere csak első
számnál
# A 2,$ a címzés megadása, $ jelenti az utolsó sort
b vege # feltétel nélküli ugrás
#zolisor cimke
:zolisor
i Zoli sor volt! # a feldolgozott sor elé beszúrás!!
:vege #vege cimke
```



The screenshot shows a PuTTY terminal window titled "szamalap.inf.elte.hu - PuTTY". The command "cat osztaly" is run, displaying a list of names and their scores. Then, the command "sedpelda3 osztaly" is run, which processes the file. The output shows that the name "Zoli" has been replaced by "kata" and a new line "Zoli sor volt!" has been added before the original list. The command "illeges@pandora:~/PANDORAHOME\_11264\$" is shown at the end.

```
szamalap.inf.elte.hu - PuTTY
illeges@pandora:~/PANDORAHOME_11264$ cat osztaly
geri 2 3 4 5 3
zoli 3 4 2 5 4
juli 4 1 2 3 3
illeges@pandora:~/PANDORAHOME_11264$ sedpelda3 osztaly
geri 2 3 4 5 3
Zoli sor volt!
kata 3 4 2 5 4
juli 4.jegy4 1 2 3 3
illeges@pandora:~/PANDORAHOME_11264$
```

# SED leírások

- A korábbi diák nem adnak teljes leírást!
- GNU SED leírás- teljes referencia
  - <https://www.gnu.org/software/sed/manual/>
- Tankönyvtár sed leírás:
  - [http://www.tankonyvtar.hu/hu/tartalom/tamop412A/2010-0011\\_szamalap1/lecke8\\_lap6.html](http://www.tankonyvtar.hu/hu/tartalom/tamop412A/2010-0011_szamalap1/lecke8_lap6.html)
- Internet ... még több leírás!

# Mi van a sed után? AWK

- A sed nagyon jó, nagyon hasznos, de ...
  - Jellemzően mintacserére használható.
  - Sorokon belüli tevékenység korlátozott.
  - Nincs aritmetikai lehetőség. (Valami van csak nem az igazi...)
  - Vezérlési szerkezetek hiányoznak. (Van ugró, feltételes ugró utasítás...,)
  - ...
- A kiút: AWK

# AWK

- Alfred V. Aho, Peter J. Weinberger, Brian W. Kernighan
- Shell hiányosságai szövegfeldolgozáskor
- Gyakorlatilag C nyelvi lehetőségek
- Tipikus szűrő
- Gyakran shell script elemként használt
- Soronkénti szövegkezelés, végrehajtódó program
- awk –gawk (GNU AWK)
  - Referencia:  
<http://www.gnu.org/software/gawk/manual/gawk.html>

# AWK működése, szerkezete

- Parancs helye: whereis awk # /usr/bin/awk, ...
- Az awk vagy paraméterként vagy a szabvány bemenetén várja az átdolgozandó adatokat.
- Soronkénti feldolgozás. Első sor előtti, utolsó utáni inicializálási blokk.
- A parancsblokkok {} jelek közti utasítás
- Parancsblokk előtt minta definiálható: Példa: /f.\*/
  - /reguláris kifejezés/
  - A minta egy logikai kifejezést tartalmaz: \$2 == „alma”

# AWK használata

- Program, közvetlenül mintegy paraméter
  - awk '{ print ;}' adatfile
    - A program minden sorra vonatkozik, kiírja azt
- File-ban a program
  - awk -f programfile adatfile
  - Helyette gyakran az awk programfile a parancs!
    - #!/usr/bin/awk -f
    - Ez az első sor paranca.
    - pl: \$ awk\_program1 adatfile # jellemző parancs alakja
- Szűrőként
  - Parancs1 | awk-parancsfile

# Bemeneti sorok elemei

- \$0 – a teljes sor
- \$1, \$2, ... - a sor első, második eleme
- Mezőelválasztó: FS (alap: helyköz vagy tab)
  - Lehet több karakter is: FS=„abc”
- Egy sor mezőinek száma: NF
- Sorelválasztó karakter: RS (alap: újsor)
- Az eddig beolvasott sorok száma: NR
  - Több bemeneti fájl esetén: FNR, egy fájl sorszáma

# AWK példa

- Programkód:

```
#!/usr/bin/awk -f
# Kezdő blokk!
BEGIN {
    print "Kezdem a programot!";
    # ide jöhetsnek a további kezdő, először
    # végrehajtandó parancsok, pl mezőelválasztó
    FS=":"
}
# minden sorban keresi a geri-t, ha találja kiírja
/geri/ {
    print "A Geri sor tartalma: "$0;
}
# Befejezés
END {
    print "Itt a vége!";
}
```

- Futtatás:

```
$ cat osztaly|elsoawk
Kezdem a programot!
A Geri sor tartalma: geri 2 3 4 5 3
Itt a vége!
$
```

# AWK kimeneti sorok elemei

- print utasítás: pl: print \$0 #teljes sor kiírása
- OFS változó
  - Output Field Separator
- ORS változó
  - Output Row Separator

```
BEGIN {  
    OFS=“:”;  
    ORS=“Vége.\n”;  
}  
{  
    print „Alma”, $0;  
}
```

# AWK változók, kifejezések

- Beépített változók nagybetűsek! Pl: NF
- Nincs típus: név=érték, érték lehet:szám, „szöveg”
  - v="alma"; print v; #alma
- Szövegösszefűzés: nincs operátor, egymás után kell írni a változókat!
  - Pl: {v="alma";f="fa"; print "5 "v f;} #5 almafa
- Konverzió automatikus, ha úgy érzi, hogy kell:
  - {v="alma";f="fa";print v+f} #0
  - {v="3alma";f="2fa"; print v+f} # 5

# AWK tömbök

- $t[0]=3$ , stb. megadjuk az index elemeket.
- Tömb elemei különböző típusúak lehetnek.
- Valójában asszociatív tömbök:
  - `t[„egy”]=1; print t[„egy”];`
- Tömb hossza: `length(t)`
- Egy index bent van-e a tömbben: `4 in t`
- Elem törlése: `delete t[4]`
- Többszemélyes tömbök: `tt[1,2]=3;`

# AWK műveletek, függvények

- +, -, ++, --, \*, /, %, ... - szokásos műveletek
- Logikai érték mint C-ben
- !=, ==, <, > - logikai operátorok
- && logikai és, || logikai vagy, ! tagadás
- ~, !~ minta illeszkedés, nem illeszkedés
  - A jobb oldali operandus lehet reguláris kifejezés is /.../.
  - A \$0 ~ /reg.kif/ alak rövidítve: /reg.kif/ , emiatt szerepel az awk blokkok előtt csak így!
- \*\* vagy ^ hatványozás, pl: 2\*\*3 #8

# AWK matematikai függvények

- Fontosabb beépített függvények:
  - int(szám) # egészrész , print int(3.7) #3
  - sqrt(szám) # négyzetgyök
  - sin(x), cos(x) # sin, cos függvények, x radiánban van!
  - rand() # ]0,1[ intervallumban véletlenszámot generál
    - x=int(10\*rand()) # 0,1,...9
  - exp(x), log(x) # e \*\* x, ln(x)
  - atan2(x,y) # arc. tangens x/y

# Egyéb hasznos awk függvények

- {v=length("almafa");print v} #szöveg hossza
- split – szöveg darabolás
  - Pl: split(„ali:pali:robi”, n, „:”); print n[1] #ali
- sprintf(sz, „minta”, változók); # mint C-ben az sprintf
- system(„parancs”) – op. rendszer par. futtatás
  - Pl: {system("date >datum") }
- File olvasás:
  - getline </home/adat”; print \$0;
  - Következő getline, következő sor.

# AWK vezérlési szerkezetek

- Elágazás
  - `if (x % 2 == 0) print "x páros,; else print "x páratlan";`
- Többirányú elágazás
  - `Switch (c) { case „a”:{... }; # mint C-ben`
- Ciklusok, mint C-ben
  - `while (kif) utasítás`
  - `do ... while (kif)`
  - `for(kif1;kif2;kif3) utasítás`
  - `for (i in tömb)`  
`tömb[i] feldolgozása`

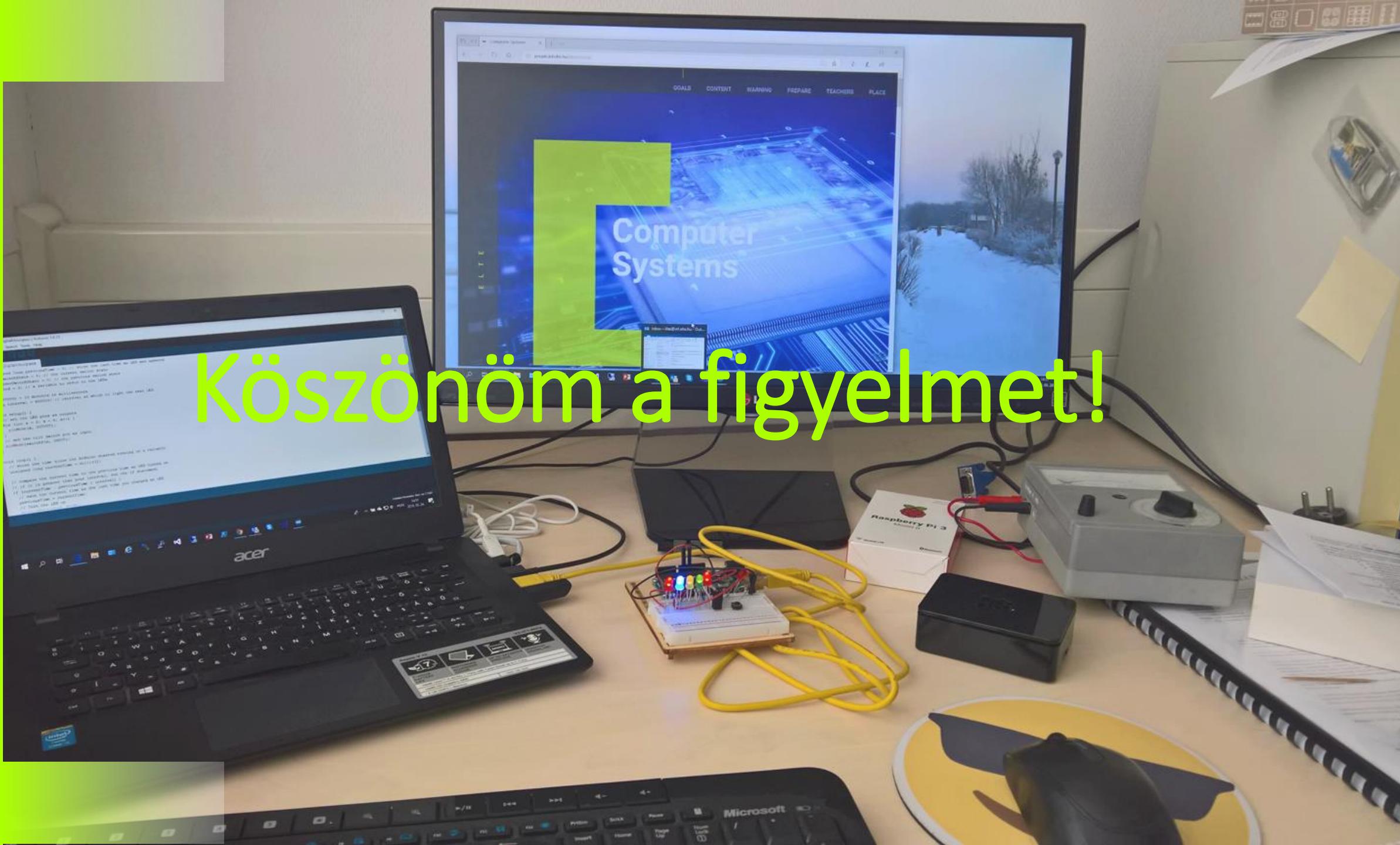
# Példa

```
#!/usr/bin/awk -f
#
# feladat: a tanulók átlagának meghatározása
# hívása: awk_atlag osztaly
# BEGIN minta először végrehajtódik
BEGIN {
    print "Ciklus-elágazás használata!";
    if (ARGC !=2)
    {
        print "Adjon meg egy fájlnevet";
        exit 1;
    }
}
```

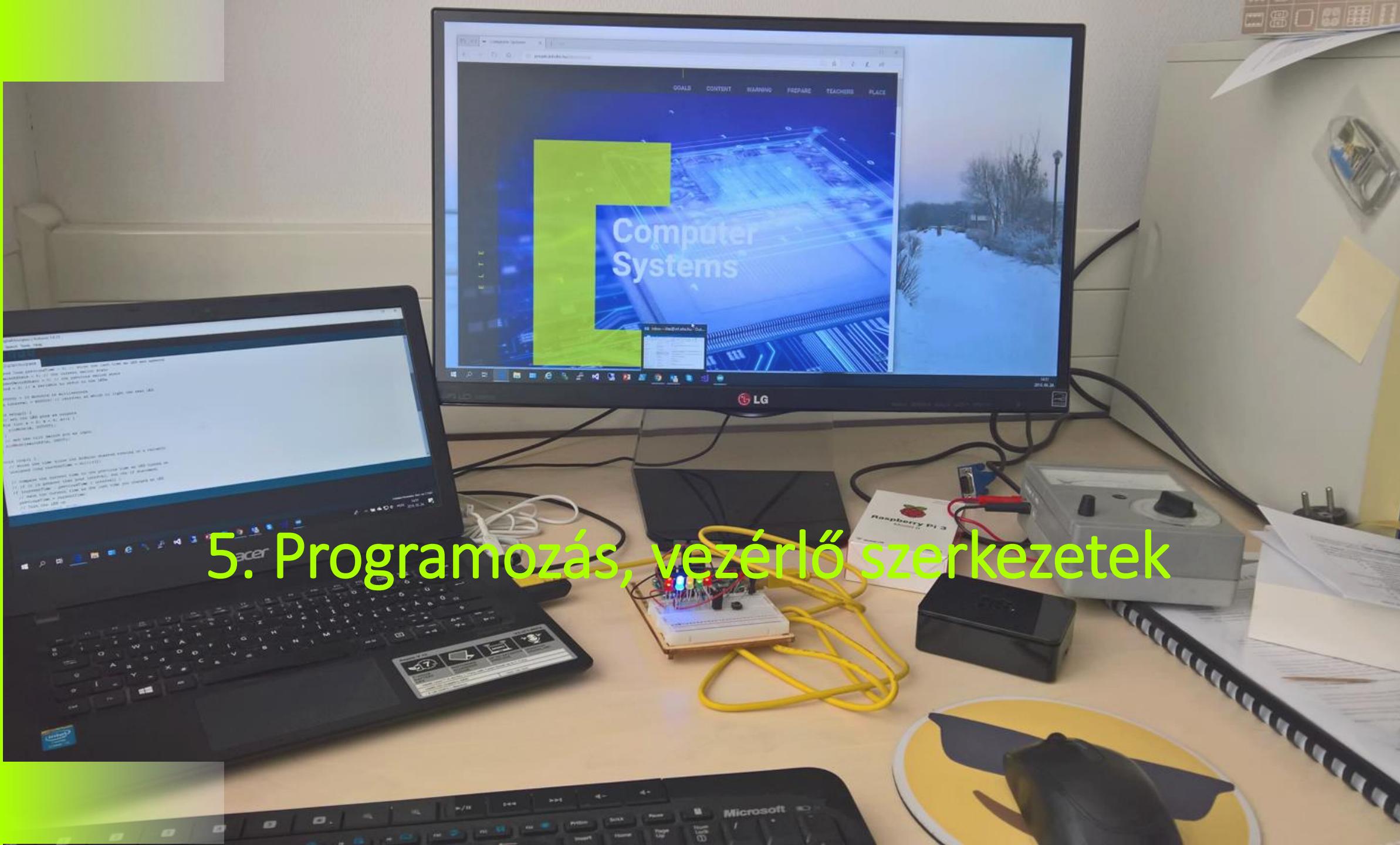
```
{
    print "A tanuló adatai:" $0; # kinyomjuk az aktuális sort
    # tanuló átlag kiszámolása
    # egy tanuló jegyeinek összeget kiszámoljuk
    osszeg=0; # kezdőérték 0
    for (i=2;i<=NF;i++) # végigvesszük a jegyeket
    {
        osszeg+=$i; # osszeghez adjuk az aktuális jegyet
    }
    # i. átlagot megjegyezzük
    atlagok[NR]=osszeg/(NF-1);
    print $1 " jegyeinek átlaga: " atlagok[NR];
}
# END minta a végén fut le
END {
    print NR " tanuló átlagát határoztuk meg!";
}
```

# AWK összegzése

- Help: Google awk, gawk
- BEGIN blokk, a soronkénti feldolgozás előtt hajtódik végre
- END blokk, a soronkénti feldolgozás után hajtódik végre
- Minta {soronkénti blokk}
- Ez a diasor a parancs legfontosabb elemeit foglalta össze!
- Teljes referencia:  
<https://www.gnu.org/software/gawk/manual/>



Köszönöm a figyelmet!



## 5. Programozás, vezérlő szerkezetek

# 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

```
utasítások  
then  
utasítások  
else  
utasítások  
fi
```

```
if  
[ $x -lt 10 ]  
then  
echo Kisebb mint 10  
else  
echo Nagyobb  
fi
```

- Nem kötelező a fentiek szerinti tagolás, csak javasolt!

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

# Ciklusok shellben: FOR

- **for** változó **in** adatlista
  - **do**
    - utasítások
  - **done**
- **for i in `who`**

```
do
echo $i
done
```
- **for i in alma körte barack**

```
do
echo $i
done
```

  - # a for ciklus általános alakja
  - # a leggyakrabban használt forma
  - # klasszikus for
  - # sorban kiírja a „gyümölcsöket”

# FOR példa - seq

- `for i in 1 2 3 4 5; do echo $i; done` # „ötös” ciklus
- Hogy készítünk N-szer végrehajtódó 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
  - do
    - 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 fájlnév, 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!

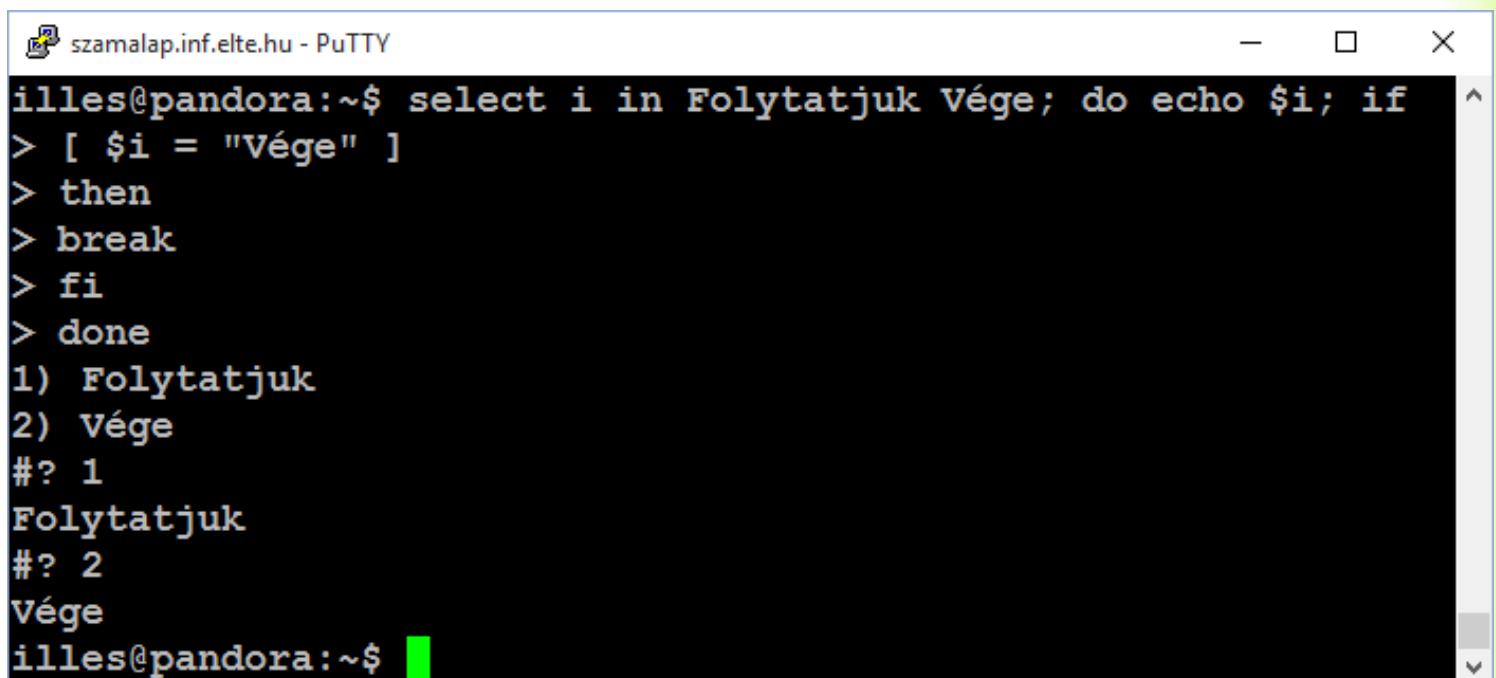
# Ciklusok shellben: UNTIL példa

- 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      # ha először nem létezik
echo Folytassuk? \(igen/nem\) # akkor létrejön a fájl
                                # Fontos a \
                                #
read v
done
```

# 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

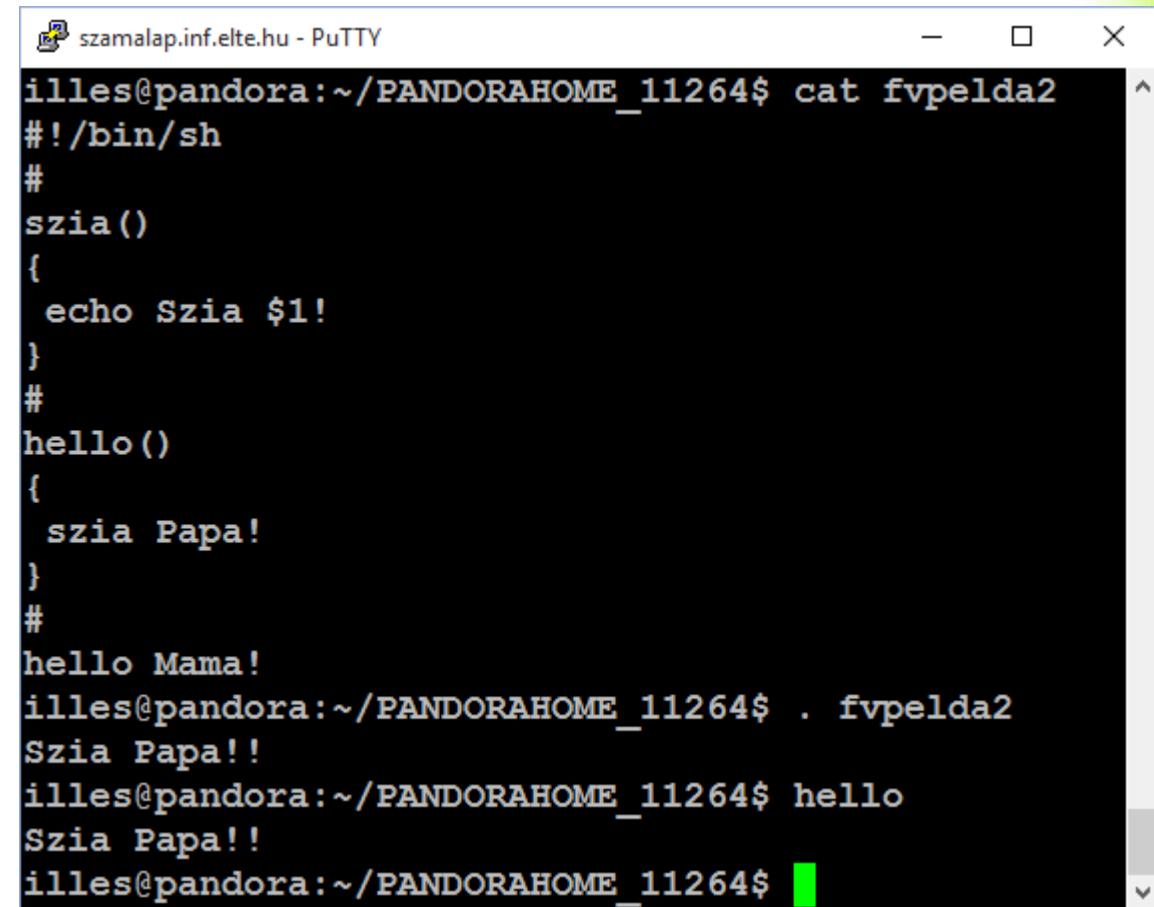
# Függvény definiálás shellben I.

- C stílusú függvény, paramétereit a shell scripthöz 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



The screenshot shows a PuTTY terminal window titled "szamalap.inf.elte.hu - PuTTY". The command "cat fvpelda2" is run, displaying a shell script. The script contains two functions: "szia()" which prints "Szia \$1!", and "hello()" which prints "Szia Papa!". The "hello" function calls the "szia" function with "Papa!" as an argument. The script concludes with a "#!/bin/sh" shebang line. Below the script, the command ". fvpelda2" is run, followed by "Szia Papa!!". Finally, the command "hello" is run, also resulting in "Szia Papa!!". The terminal prompt "illes@pandora:~/PANDORAHOME\_11264\$" is visible at the bottom.

```
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 szamolvas
  - A végrehatási szál utasításait írja ki. A -v a teljes elágazást kiírja, míg a -x a tesztelt logikai kifejezéseket írja csak.
- Sh -n szamolvas
  - 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 ./fájlnév
# 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
```

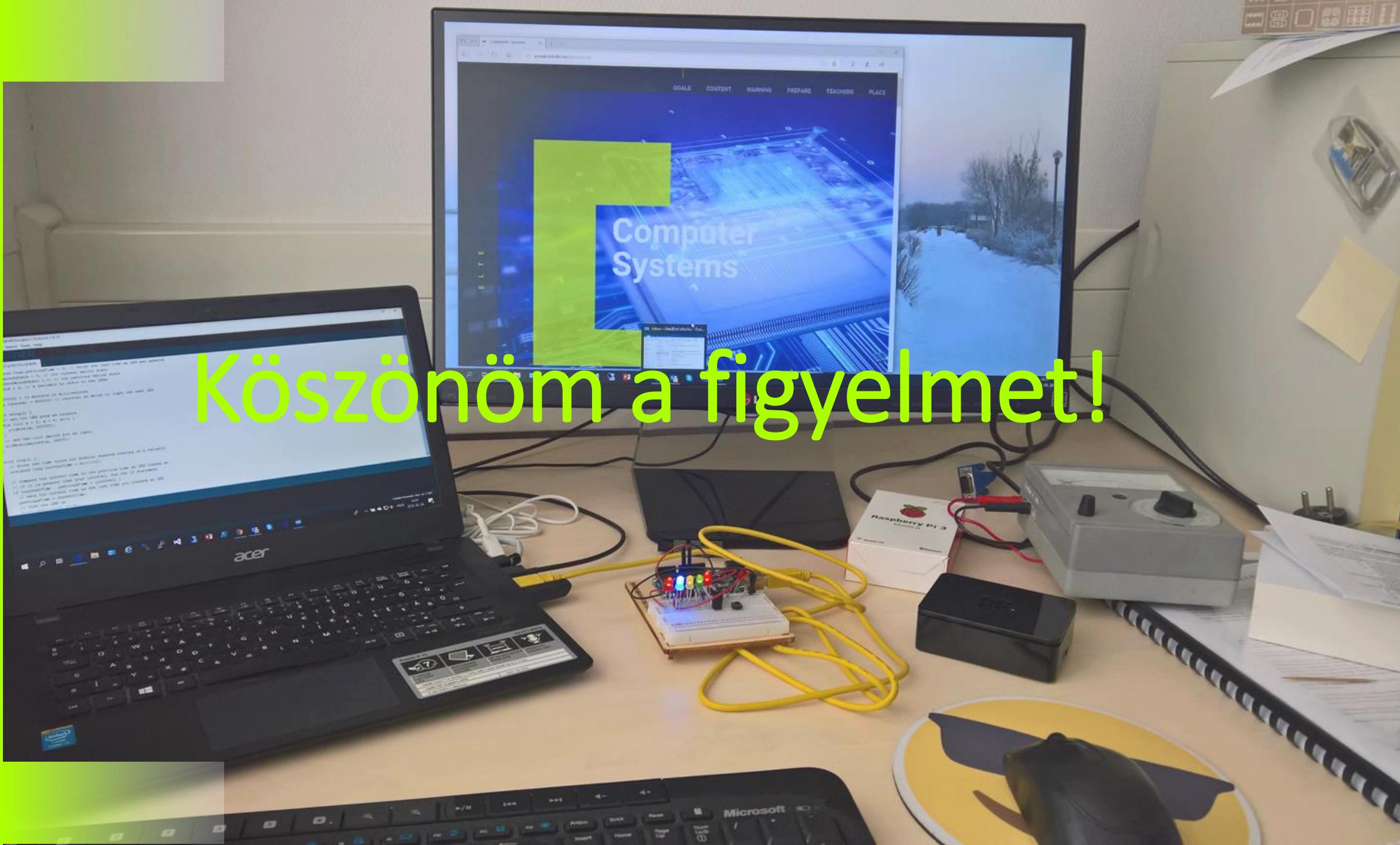
# Csomagoló használat

- A csomagoló script az eredményét a standard outputra írja!

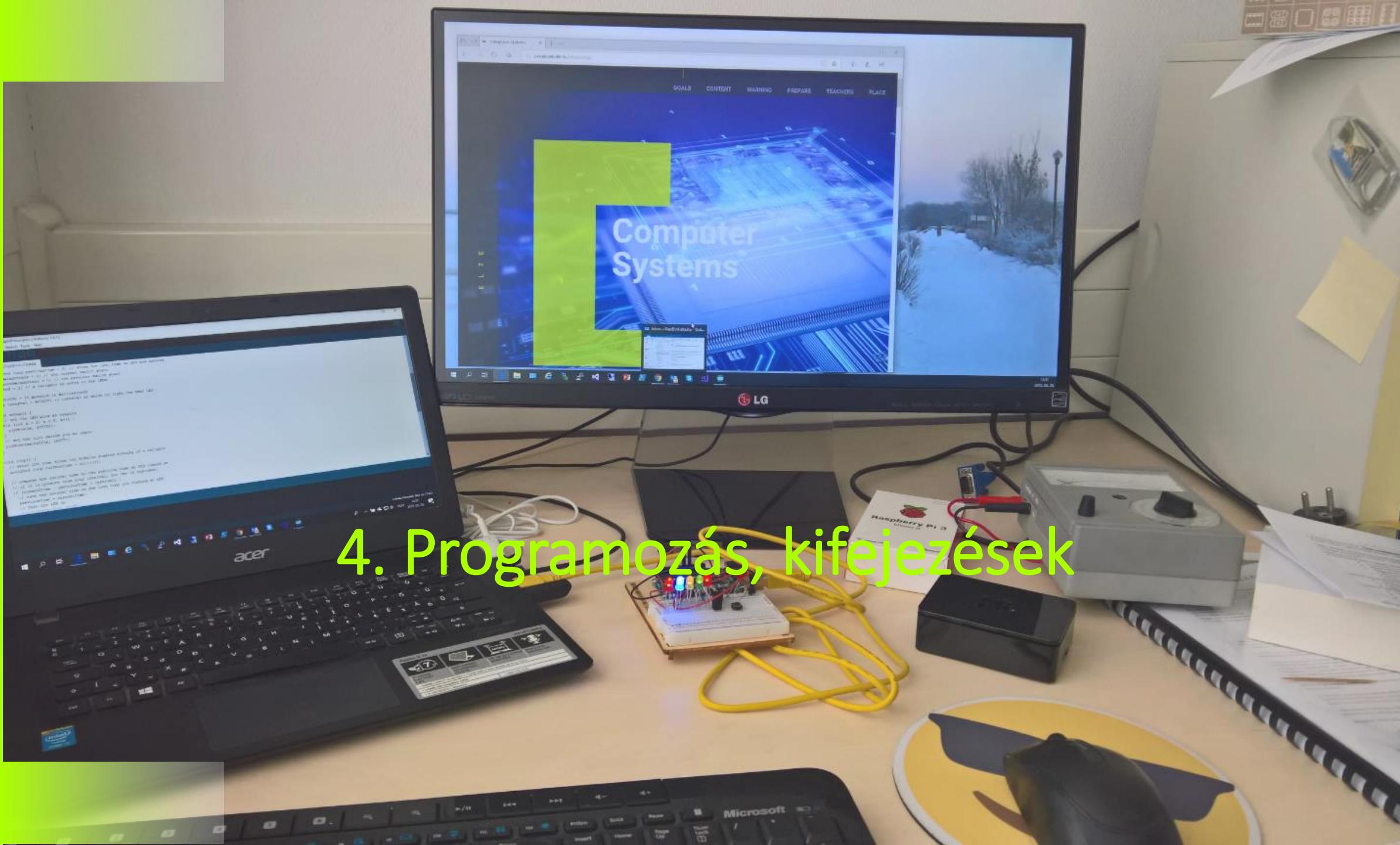
```
$ csomagol forpelda valogat # eredmény a képernyőre
$ ....
$ csomagol forpelda valogat >csomag #eredmény fájlba
$ sh ./csomag      # kicsomagolás, mivel nincs x jog
                      # ezért így kell indítani
```

# Demo...

- Csomagol ...



Köszönöm a figyelmet!



## 4. Programozás, kifejezések

# Visszatekintés

- Számítógépek, jelek, információk, számábrázolás, kódolás
- Felépítés, operációs rendszer, kliens-szerver szerep
- Grafikus, karakteres kapcsolat, fájlrendszerek
- Alapvető parancsok, folyamatok előtérben, háttérben
- I/O átirányítás
- Szűrők
- Reguláris kifejezések

# Mi jön ma?

- Változók, használatuk
- Változó behelyettesítés
- Parancs behelyettesítés
- Elemi műveletek (aritmetikai, szöveges)
- Logikai kifejezések
- ...

# Változók UNIX shellben I.

- Szöveges változó létrehozására van lehetőség.
  - Környezeti vagy shell változóról beszélhetünk.
- Változónév betű lehet, majd szám és aláhúzás karakterek használhatók.
- Környezeti változók, mind a környezetben, mind az abból indított parancsokban láthatók. (globális)
  - env – program futása a definiált környezetben
    - Ha nincs paraméter akkor a környezeti változókat írja ki!
    - Különböző rendszerekben eltér a kiírt változók köre!
  - PATH (.profile), elválasztó karakter :, PS1, LOGNAME, stb.

# Változók UNIX shellben II.

- Változó definiálás: = jel
  - Példa: csapat=Fradi; szam=5;
  - mondat="Alma a fa alatt."
  - spec\_mondat='Ez 5\$ és 3%\10% lesz!'
- Változó tartalma mindig karakter, még a szám se szám!
- set parancs: összes változó kiírása (környezeti is)
- unset parancs: változó törlése
  - unset csapat # set eredmény: \_=alma

# Változó tartalma, hatóköré

- Példa: hajra="Hajrá Fradi!"
- Tartalom: \$hajra
  - echo \$hajra # Hajrá Fradi!
  - echo Mondjuk: \$hajra #behelyettesítés
- export hajra: A hajra változó környezeti lesz!
  - Ettől az unset ezt is törli!
- Környezeti változót mindenki látja!
  - Sima változót csak az aktuális shell!

# Változó behelyettesítés

- csapat=Fradi; echo \$csapatka # Mi lesz az eredmény?
- echo \$csapat a legjobb!; # Természetesen!
- echo \${csapat}a legjobb!
- unset csapat; x=\${csapat-Újpest} # ha csapat nincs, x=Újpest
  - inicializálás ha nem létezik
  - x=\${csapat=Újpest} # csapat is változik!
  - echo \$x a legjobb! # ???
- y=\${csapat?Szia} # ha csapat nem definiált, a Szia kiírásra kerül, majd kilép a shell, y nem kap új értéket!
- y=\${csapat+Újpest} # ha csapat definiált, y=Újpest

# Parancsbehelyettesítés

- `parancs` parancsot végrehajtja, parancs kimenete kerül a helyére
  - Bash shellben: \$(parancs)
  - ki\_vagyok=`whoami` # \$(whoami)
  - a=`date` ; b='date' #
  - echo \$a # ???
  - echo \$b # ???
    - A date szó lesz az eredmény.
  - eval \$b #man eval,
    - Ugyanaz az eredmény, mintha a parancs \$b lenne eval nélkül!

# Első shell scriptem

- A fájl neve: else
- Egy kis módosítás:

```
illes@panda:~$ vi else  
illes@panda:~$ chmod +x else  
illes@panda:~$ cat else  
echo Ez az első shell scriptem!
```

```
illes@panda:~$ ./else  
Ez az első shell scriptem!  
illes@panda:~$ else  
Ez az első shell scriptem!  
illes@panda:~$
```

```
illes@panda:~$ cat else  
#!/bin/sh  
#  
echo Ez az első shell scriptem!
```

# Műveletek shellben

- Egyetlen művelet létezik: szöveg összefűzés
  - Shellben minden változó szöveg!
- `x=piros; echo Nyári $x alma! # ?`
  - Aritmetikai műveletek közvetlenül nem használhatók.
  - `x=alma; y=$x+fa; echo $y # alma+fa`
  - `a=5; b=$a+1; echo $b #5+1`
- A szövegösszefűzésen kívül közvetlenül se egyéb szöveges, se aritmetikai műveleteket nem támogat!

# Aritmetikai műveletek

- let utasítás, a bash része, régi sh-ban nincs
  - a=2; let b=a+1; echo \$b # eredmény 3
- expr utasítás
  - expr \$a op \$b PL: expr 3 \\* 4 !!!
  - op: <,<=,>,>=,!=,=,+,-,\*,/,% (mod)
  - Javasoljuk az expr használatát kompatibilitás miatt!
- bc parancs (szűrő)
  - C nyelv jellegű bemeneti szöveget vár
    - Létezik ciklus, szögfüggvények, fv definíció, stb
  - Példa: echo 2\*16 | bc #32

# BC példa

- Összetettebb példa:
  - Függvény definíció
  - Négyzetgyök, szögfüggvény használatra példa!
- Fájl név: bcpelda
- Futtatás: chmod +x bcpelda
  - ./bcpelda

```
#!/usr/bin/bc -lq
#a -l kapcsoló a math könyvtárat használja
# ez kell s szinusz fv-hez (s)
define fakt (x) {
    if (x <= 1) return (1);
    return (fakt(x-1) * x);
}
print "Az 5 faktorialis erteke: ";
print fakt(5);
print " !\n";
print "A 25 négyzetgyöke: ";
print sqrt(25);
print "\n";
print "Színusz PI/2 értéke:";
print s(3.1415/2);      #színusz
print "\n";
quit
```

# Parancs paraméterek

- Példa: legjobb csapat a Fradi! #ez parancs ☺
    - Mi a parancsnév?
  - \$1, \$2, \$3, ... # paraméter változók
    - echo \$1 # csapat
  - \$0 # parancs neve (legjobb)
  - \$# # paraméterek száma
  - \$\* # összes paraméter, idézőjel hatás nincs!
  - „\$@” # külön a paraméterek, példa: param

# Paraméter példa

- Hívása: param barack fa 'alma fa'

```
echo paraméter használat
echo "adunk összetett paramétert is 'alma
fa'"
# minden külön
echo '$*' használat'
for i in $*
do
echo $i # barack, fa, alma, fa külön sorban
done
```

```
# minden egyben
echo "$*" használat for ciklusban'
for i in "$*"
do
echo $i # eredmény egy sorban!
# "között" vannak a paraméterek
done
echo "$@" használat'
# az alábbi sor a for i in $@ alak rövid változata
for i
do
echo $i
done
```

# Param példa futása

```
illes@panda:~$ param fű fa 'alma fa'  
paraméter használat  
adunk összetett paramétert is ('alma fa')
```

```
$* használat  
fű  
fa  
alma  
fa  
"$*" használat for ciklusban  
fű fa alma fa  
"$@" használat  
fű  
fa  
alma fa  
illes@panda:~$
```

# Még több parancs paraméter

- Ismétlés:
  - \$0 – parancs név
  - \$1,...\$9 paraméterek
- Csak 9 paraméter lehet?
- Nem, használhatjuk a {} változó megadó karaktereket!
- Igaz, elég ritka a nagy paraméterszám!
  - Nem ajánlott a túl sok (4 vagy több) paraméter használat!

# Nagyszámú paraméter használat

- \${10}
  - Tizedik paraméter, zárójelek nélkül \$10, a \$1-hez fűzné a 0 karaktert!
  - \${100} # 100. paraméter...
- Shift utasítás
  - A paramétereket eggyel balra lépteti.
    - Ez \$1-et kidobja, majd balra lépnek egyet a paraméterek,\$2->\$1,... \$10->\$9 (\$10 –be \$11 kerül, ha van)
  - Jellemző feldolgozás ciklus segítségével.

# Egyéb hasznos változók

- \$\$ - A futó program PID értéke!
- \$! - A háttérben utoljára végrehajtott program PID-je!
- \$- - A shell kapcsolói . ?????

# Program befejezése, eredménye

- Egy shell script program az utolsó utasítás elvégzésével befejezi működését, visszatér a hívó félhez, a shellbe!
- Van-e ennek eredménye?
  - Igen!
- minden utasításnak van befejezési eredménye!
  - Ez az eredmény a program sikerességét, sikertelenségét mutatja!
- Egy program sikeresen lefut, ha végre tudja „rendesen” hajtani a feladatát!
  - Ez programonként más és más!

# Parancsok eredménye

- Ilyen parancs eredmény minden esetben létrejön!
- A parancs eredményét a \$? változó tartalmazza!
- echo szia! ; echo \$? # 0 lesz az echo parancs eredménye!
- Ha az eredmény 0, akkor a parancs sikeresen lefutott!
- Ha pozitív az eredmény (legtöbbször 1), a parancs sikertelen

```
illes@panda:~$ cp beka peti
cp: stat "beka" sikertelen: Nincs ilyen fájl vagy könyvtár
illes@panda:~$ echo $?
1
illes@panda:~$
```

# exit parancs

- Közvetlenül az exit parancs segítségével ki is léphetünk egy shell programból!
- exit érték # ahol érték egy 0-255 közti egész lehet
  - Ha az exit paramétere 0, az azt jelenti, hogy a befejezett program sikeres volt!
    - Ez a sikeresség **logikai igazként** is értelmezhető!)
  - Ha a paraméter pozitív, a futás sikertelen!
    - Ez a sikertelenség **logikai hamisként** is értelmezhető!
- Hasonlít a C nyelvhez, csak ott fordított az értelmezés!

# Logikai kifejezések

- Lássuk az alábbiakat:
- Látható: Nincs önmagában logikai kifejezés!
- Értékadás van, ez mint logikai is értelmezhető! (igaz)
- Csak a [] logikai kifejezés jó!

```
szamalap.inf.elte.hu - PuTTY
illeg@panda:~$ if alma=barack; then echo azonos; else echo nem; fi
azonos
illeg@panda:~$ x=alma
illeg@panda:~$ if (alma=barack); then echo azonos; else echo nem; fi
azonos
illeg@panda:~$ if ($x=barack); then echo azonos; else echo nem; fi
-bash: alma=barack: parancs nem található
nem
illeg@panda:~$ if ($x==barack); then echo azonos; else echo nem; fi
-bash: alma==barack: parancs nem található
nem
illeg@panda:~$
```

```
szamalap.inf.elte.hu - PuTTY
illeg@valerie:~$ if [ alma = barack ]; then echo azonos; else echo nem azonos; fi
nem azonos
illeg@valerie:~$
```

# Logikai eredményt adó utasítás

- Ahogy láttuk, logikai művelet nincs!
  - De minden utasítás eredménye logikai eredményként is értelmezhető!!!!
- Segít a **test** utasítás! Ezt az utasítást gyakran a [] karakterekkel helyettesítik!
- **test operandus1 operátor operandus2 #** fontos a helyköz!
  - Vagy: [ operandus1 operátor operandus2 ] # itt is fontos a helyköz, [ után és ] előtt is!

# Test – aritmetikai műveletek

- test, vagy [ ... ] logikai vizsgálat
- 0 – igaz, 1- hamis, echo \$?
- true – igaz parancs, exit 0
- false – hamis parancs, exit 1
  - -lt,-gt.-le,-ge,-eq,-ne numerikus vizsgálat
    - [ \$x -lt 5 ]
    - -f file, -d dir file vagy könyvtár létezés
    - -o, vagy, -a az és operátor, ! a tagadás
    - -r,-w,-x fájl read,write,execute jog megléte

# Test – szöveges, fájl vizsgálat

- Szöveg összehasonlítás:
  - `=`, `!=` azonos, nem azonos sztring vizsgálat
    - `test $a = $b` # igaz, ha \$a és \$b azonos szöveg
    - `test $a != $b` # igaz, ha \$a és \$b nem azonos szöveg
  - `test -z $X` # igaz, ha \$X szöveg hossza 0, üres sztring
  - `test -n $X` # igaz, ha \$X string hossza nem 0, ha nem üres
    - `[ -n $X ]` # fontos a helyköz!!!!
- Teljes referenciához: `man test`

# Test példa – I.

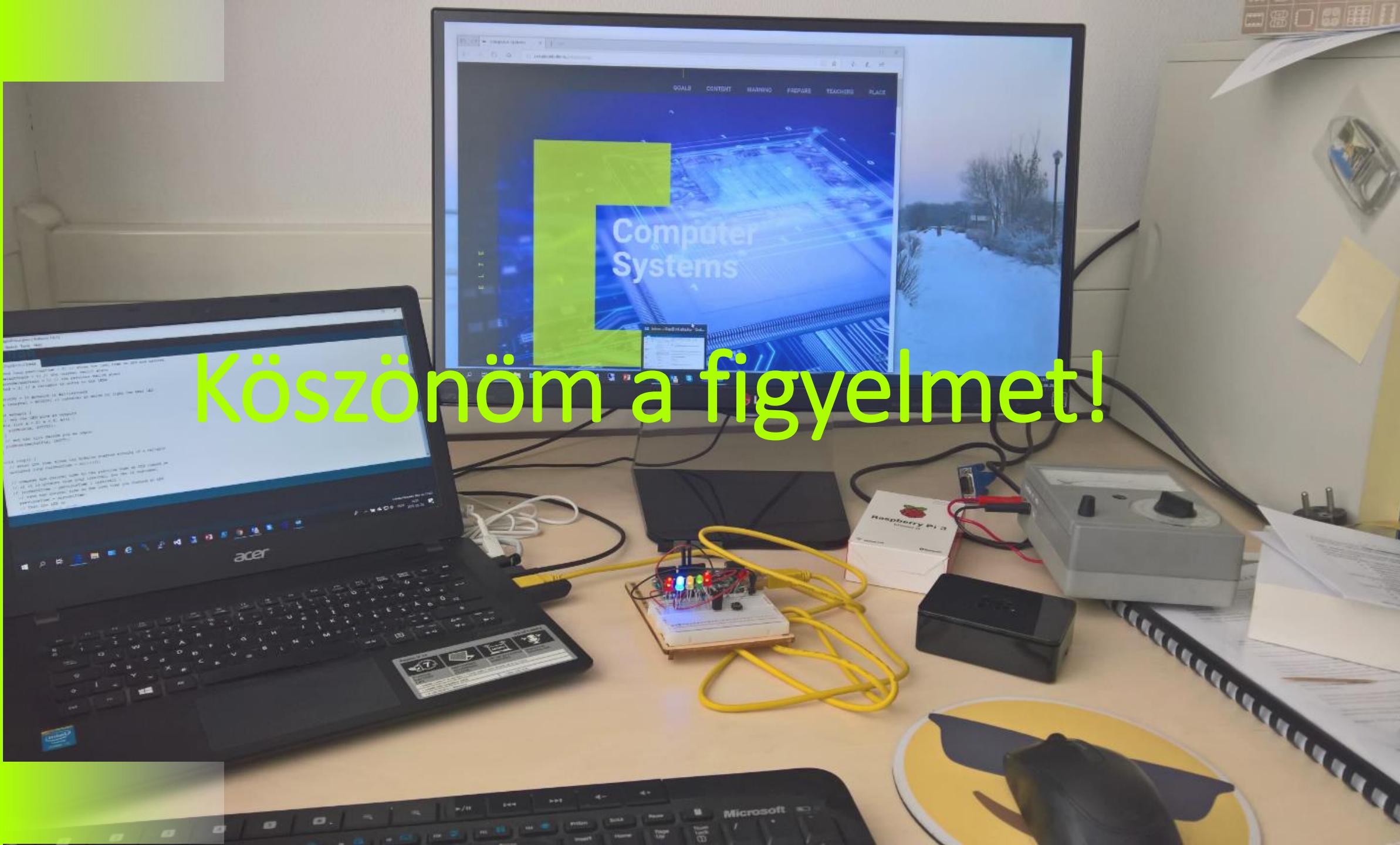
- A példa a leggyakrabban használt jellemzőket mutatja!
- Egyszerű logikai kifejezések.

```
$ x=4  
$ [ $x -lt 6 ] # test $x -lt 6  
$ echo $?  
0 (igaz)  
$ test "alma" = "körte" # [ „alma” = „körte” ]  
$ # szöveges azonosság vizsgálat eredménye  
$ echo $?  
$ 1 (hamis)  
$ test -z „alma” # üres string vizsgálat  
$ echo $?  
1
```

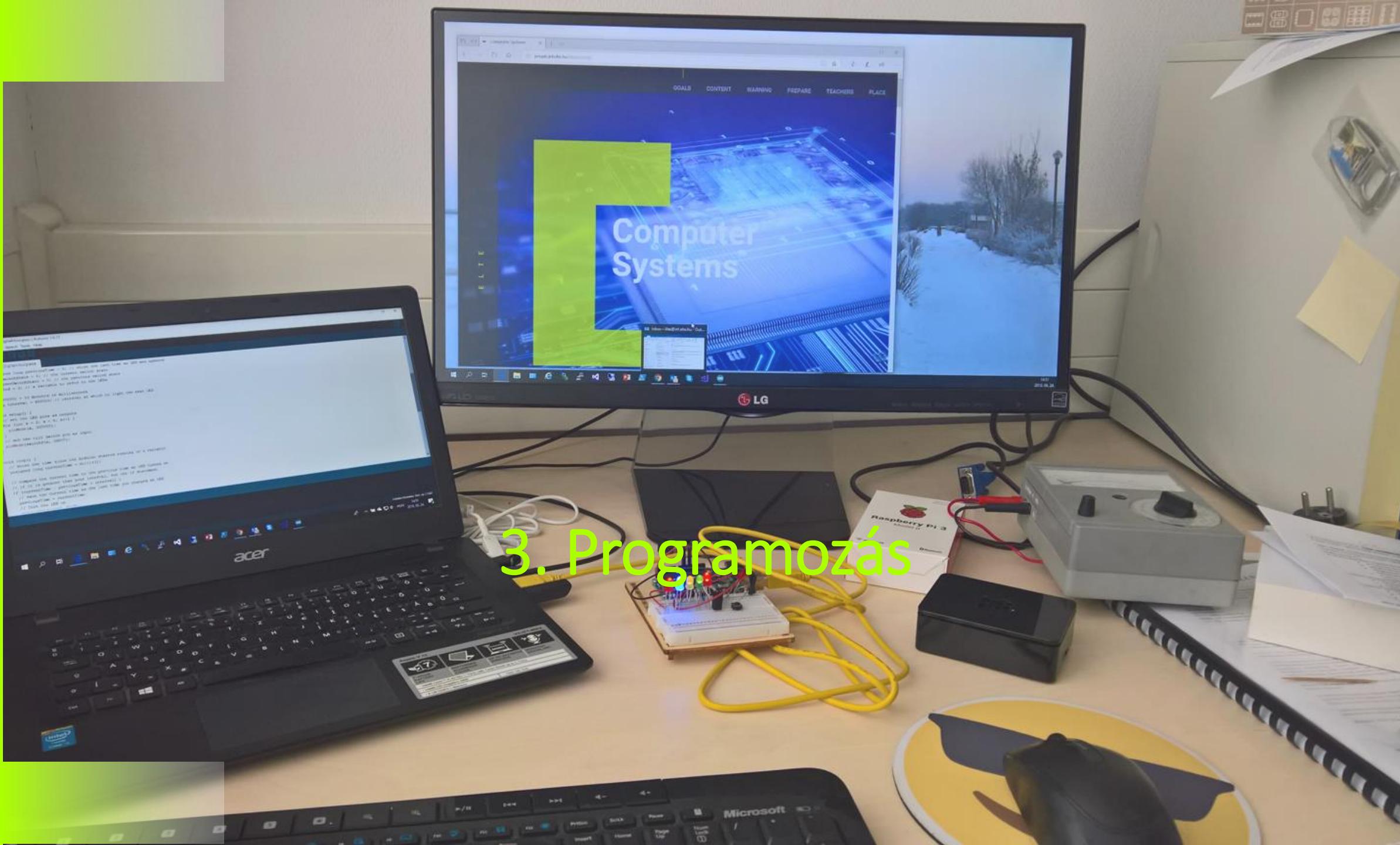
# Test példa – II.

- Összetett logikai vizsgálat
- Fájl, könyvtár vizsgálat

```
$ x=4  
$ y=fradi  
$ [ $x -eq 4 -a $y != „vasas” ]  
$ # logikai ÉS kapcsolat -a  
$ echo $?  
0 (igaz)  
$ [ ! $x -eq 4 ]  
$ echo $?  
1 (hamis)  
$ test -f fradi # igaz, ha fradi fájl létezik  
$ [ -d alma ] # igaz, ha az alma könyvtár  
# létezik
```



Köszönöm a figyelmet!



# Visszatekintés

- Számítógépek, jelek, információk, tárolásuk
- Számábrázolás
  - Fixpontos, lebegőpontos számábrázolás
- Kódolás
  - ASCII, UTF-8 stb kódtáblák
- Felépítés, operációs rendszer szerep
- Kliens-Szerver különbségek
- Grafikus, karakteres kapcsolat
- Fájlrendszer, alapvető parancsok

# Mi jön ma?

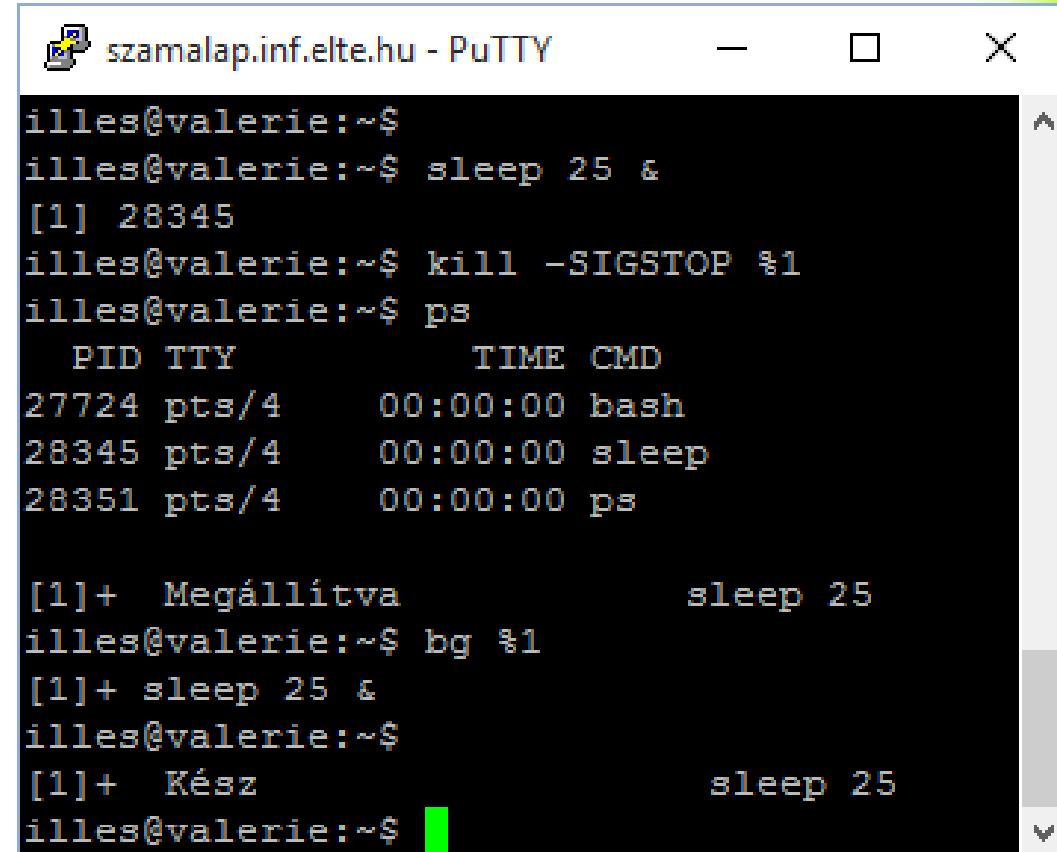
- További parancsok
- Folyamatok előtérben, háttérben
- I/O átirányítás
- Szűrők
- Irány programozni!
- Reguláris kifejezések

# További parancsok

- Parancsok előtérben: normál parancs kiadás
- Parancs futtatás a háttérben: & karakter a parancs végén
  - & megnevezései: ampersand, and, és jel, at jel
- sleep 15 # 15 másodperc várakozás, közben nincs parancs kiadási lehetőség
  - CTRL+Z stop jelet küld a folyamatnak
- sleep 15 & # a várakozó folyamat a háttérben fut, közben újabb parancsot adhatunk ki
  - Eredményül kapjuk: [1] 28321, az első a „jobszám”, a második a pid
- jobs – listázza a futó parancsainkat(„jobjainkat”)
- ps – process status parancs, láthatjuk a háttérben futó sleep-et is

# Folyamatok háttérben

- Mi van ha egy háttér folyamatot előtérbe akarunk tenni: fg %1
- Egy előtérben futó folyamatot ha leállítottunk(ctrl-z) a háttérben is folytathatjuk: bg %1
- Jelzés küldés: kill –signál folyamat
  - A folyamatot vagy a job sorszámaival (kell a % jel)vagy a processz azonosítóval jelöljük.



The screenshot shows a PuTTY terminal window titled "szamalap.inf.elte.hu - PuTTY". The session ID is "illes@valerie:~\$". The terminal displays the following command-line session:

```
illes@valerie:~$ sleep 25 &
[1] 28345
illes@valerie:~$ kill -SIGSTOP %1
illes@valerie:~$ ps
  PID TTY          TIME CMD
27724 pts/4    00:00:00 bash
28345 pts/4    00:00:00 sleep
28351 pts/4    00:00:00 ps

[1]+  Megállítva                  sleep 25
illes@valerie:~$ bg %1
[1]+ sleep 25 &
illes@valerie:~$ 
[1]+  Kész                      sleep 25
illes@valerie:~$ 
```

# Még többet a folyamatokról!

- top parancs: látjuk a futó folyamatok adatait, globális rendszer állapotot!
  - mindenki egyenrangú?
  - Unix rendszerek prioritása: -20-tól 19-ig (40 szint)
  - A kisebb szám a nagyobb prioritás!
  - nice parancs, módosítja az indítandó parancs prioritását.
    - Alapból a prioritás 0, top vagy ps -l parancs NI oszlop!
    - nice -n 5 sleep 20& # 5 lesz az új prioritás, 0-hoz 5-öt ad
    - nice -n -10 sleep 20& # növelni root joggal lehet!!!!

# Folyamatok, prioritások

- A Unix, Linux prioritás alapú folyamatkezelést végez!
- POSIX4 – IEEE1003.1b (1993), Valós idejű kiterjesztés megjelenés
- Két prioritás lista
  - nice -20-19, ahogy láttuk
  - Valós idejű prioritási lista, 0-99 közti értékekkel.(100 prioritási szint)
    - Ebben a listában a nagyobb szám jelenti a nagyobb prioritást
  - A lista közös értelmezése a következő:
    - 99,98...1,-20,-19,..0,1...,19, ezt gyakran 140-es prioritás intervallumnak neveznek, amiben különböző eltolások(mapping) lehetségesek.

# Parancsok időzített futtatása

- Unix rendszereken kétféle időzített futtatási támogatás van.
  - cron (démon), rendszergazda jogosítvány kell, vagy a /etc/cron.allow-ban engedély
    - /etc/crontab – rendszer időzítések, /etc/cron.d/ - felhasználói időzítések
    - Ezen időzítések megadják, hogy a cron milyen programokat milyen időpontokban futtasson.
  - at – a parancs használható, ha az atd szolgáltatás fut.
    - at parancs a standard inputról várja a bemenetét(a futtatandó parancsot)
    - at now + 5 minutes <parancsfájl # az at-nek az időpont kell mint paraméter, amit sokféleképpen megadhatunk, lásd man!

# Parancsok befejezése

- Normál befejezés – a parancs nem akar tovább futni!.
- Egy parancs maximum addig fut, amíg az indító felhasználó a rendszerben van!
  - Mindegy, hogy el előtérben vagy háttérben indítottuk el!
- nohup parancs
  - nohup parancs & # lehet & nélkül is, csak akkor előtérben fut!!!
    - A parancs kimenete a nohup.out állományba kerül!
    - Ez az állomány indításkor létrejön, a későbbi outputot ehhez hozzáfűzi!
    - Saját kimenet állomány is megadható: nohup parancs >sajat.nohup &

# Idézőjelek

- Parancssorban legális karakterek: .,\_,-,számok, normál karakterek
- Idézőjelek: ' ", \ – módosítják a klasszikus karakter értelmezést
- 'Aposztrof karakterek között megszűnik minden speciális \ \$ % stb. hatás'
  - pl: echo 'alma\\$fa' #alma\\$fa
- "Macskakörmön belül a \$, a \, a ` és a ' karakterek hatása megmarad"
  - Példa: a=fradi; echo "hajrá \$a !" # hajrá fradi
  - echo "Hajrá ' aposztrof!" # aposztrof karakter kiírás
- \x karakter: módosítja x eredeti jelentését
  - Példa: fa=virág; echo alma\\$fa #alma\$fa
  - echo alma\$fa #almavirág

# Kimenet, Bemenet, átirányítások

- Kimenet, bemeneti eszközök
  - stdin (0) - billentyűzet, alapértelmezett bemenet
  - stdout (1) - monitor, alapértelmezett kimenet
  - stderr (2) – monitor, alapértelmezett hibakimenet
- Átirányítás
  - Kimenet: > jel segítségével, új állomány létrejön
    - Pl: echo alma barack szilva >gyumolcs #gyumolcs új fájl lesz
    - echo alma >&2 #hogy ne a 2 nevű állomány legyen
  - Bemenet: < jel segítségével
    - Pl: passwd juli <alma; cat alma # almafa, almafa

# I/O átirányítások II.

- Kimenet, hozzáfűzés (append)
  - >> fájlnév, Pl: echo dió >>gyumolcs
  - Ha nem létezik a fájl, akkor létre is hozza!
- Hibakimenet (stderr) átirányítás
  - 2>, 2>>
- Szimmetria miatt(☺): <<
  - Bemenet átirányítás a helyben megadott szövegre

```
$ cp 2>hiba  
$ mv ezt 2>>hiba  
$ cat hiba
```

```
cat <<alma  
<input type=text name=X>  
<input type=button>  
alma
```

# Szűrők

- Parancs vs. Szűrő
  - Képes egy parancs kimenetét saját bemeneteként fogadni!
- Önmagában egy szűrő nem szűrő!
  - Legalább 2 parancs összekapcsolásáról van szó!
- Műveleti jel: |
- Ilyen nagyon ismert, gyakran használt szűrőparancs például a WC!
  - Word Count!
  - Feladata: sorok, szavak, karakterek megszámolása!
    - `wc [-lwc] [file]` #Paraméter nélkül várja ctrl-d-ig a bemeneti adatokat!

# Szűrők használata

- Használhatók normál parancsok alakban, paraméter nélkül, ekkor a bemenetről (billentyűzet) vár adatokat!
  - Input vége:CTRL+D
  - Paraméteres formában módosulhat a bemenet!
  - Pl: wc alma.fa
- Szűrő formában, | karakterrel!

The screenshot shows a PuTTY terminal window titled "szamalap.inf.elte.hu - PuTTY". The session ID is "illeg@panda:/\$". The terminal displays the following command-line interactions:

- First command: "wc alma a fa alatt Esik az eső" followed by two lines of output: "2 7 29".
- Second command: "ls | wc" followed by three lines of output: "27 27 124".
- Third command: "ls" followed by a long list of directory entries:

afs	dev	h2	lib	opt	sbin	tmp
bin	etc	home	lib64	proc	srv	usr
boot	h	initrd	media	root	sys	var
cluster	h1	LDAPFETCH	mnt	run	T	
- Fourth command: "ls -l | wc" followed by three lines of output: "28 247 1338".
- Fifth command: "illeg@panda:/\$"

# Fontosabb szűrők

- Fontosabb kész szűrők:
  - tee, tr, cut, sort, uniq, wc, grep, stb.
  - Példa:

```
$ who >nevek  
$ sort nevek #sorok szerinti sorrend  
$ who|sort -r -u # fordított sorrend, egyedi sorok  
$ who|wc -l #bejelentkezett felhasználó szám
```

- Ami biztos: man (ual) lekérdezése
  - Példa: man sort, man -k kulcsszó

# Tee -, tr parancsok

- tee – a paraméterként kapott fájl(ok)ba másolja a csővezetéken áthaladó tartalmat.
  - Példa: ls -l | tee dir.txt | wc -l, who | tee -a users.txt report.txt | sort -r
    - -a paraméter: append
- tr – translate, paramétere kér karakter csoport, a szabványos bemenetre érkező adatokban lévő első karaktercsoport elemeit a második karaktercsoport elemekre cseréli.
  - Példa: echo alma | tr a e # elte
    - Hasonló lehetőség a sed y paranca! (később látni fogjuk)

# CUT - kivágás

- Standard inputon vagy fájlból vághatunk ki mező(ke)t, oszlop(ka)t, sorkból!
- cut -c1-5 1-5 karakteroszlop kivágás
  - példa: date|cut -c4-8 # Oct
- cut -f1,3,5-7 1,3,5,6,7 mezők kivágása
  - Alapértelmezett mezőelválasztó: Tab
  - Új mezőelválasztó: -d char
  - Példa: cat /etc/passwd|cut -f1,7 -d: # név, shell

# Grep – sorok szűrése

- A paraméterül adott mintával rendelkező sorok kiválasztása.
- Fontosabb paraméterek:
  - -v mintát nem tartalmazó sorok
  - -i kis és nagybetűket nem különböztet meg
  - -w Csak önálló szóként találja meg (traPista nem)
  - -r Rekurzívan a paraméterül adott könyvtárra.
  - -l Csak a fájl neveket írja ki. (fájlban keres)
  - -c csak a sorok számát írja ki
  - -n megszámozza a sorokat

# Grep használata

- Szűrő példa:
  - cat nevsor|grep Pista # Eredményül kapjuk a Pista-t tartalmazó sorokat.
- Parancs forma:
  - grep -r 'fradi' ./script # script könyvtárban a fradi-s sorokat (fájlokban) keresi
  - grep -r -l par \* # Az összes állományban, alkönyvtárakban keresi a par szót, eredményül csak a fájl nevét írja ki.

# Mintaillesztés, reguláris kifejezések

- Egy szövegminta általános megadása – Reguláris kifejezések-speciális karakterek
  - ^ Sor elejétől kell egyezni a mintának.
    - Pl: '^alma' : a sor elején alma szó áll
  - \$ Sor végétől kell egyezni a mintának.
    - Pl: 'barack\$' : a sor végén a barack szó áll
  - . Egy tetszőleges karakter
  - \* Előző minta ismétlése 0 vagy többször!
    - Pl: '^alma.\*fa\$' - alma és fa között akárhány(0 is lehet) karakter

# Példák I.

- Cat fájl|grep „^\$” #Üres sor kiválasztás
  - Hány üres sor van egy fájlban? Cat fájl|grep „^\$”|wc -l
- Cat fájl|grep „^\$ FTC.\*\\$\$” # Sor elején, végén \$, első \$ után FTC, utána bármi!
- Cat fájl|grep „/-” # Valahol a sorban /- karakterek.
- Cat fájl|grep „-/-” # Hiba, mert – jelet parancs kapcsolónak tekinti, és nincs / jelű kapcsoló!
- Cat fájl|grep „\-/-” # Ez már OK, a -/ karakterek bárhol szerepelhetnek!

# További mintaillesztés

- Karakterhalmazok megadása: [FTC] # Vagy F vagy T vagy C
  - Karakter intervallum megadása(- jel): [a-z] (kisbetű)
  - [^a-c] Nem a, b vagy c betű
  - [A-Za-z0-9] Alfanumerikus (szám vagy betű)
    - \w Alfanumerikus, mint előző
    - \W Nem alfanumerikus
  - \d számjegy, azonos a [0-9]-el
  - \s szóköz, tab, sortörés
- Szavak illesztése
  - \< Szókezdet, Pl: grep "\<Zol"
  - \> Szóvég, Pl: grep "tán\>"
  - \b Szó elején, végén helyköz Pl: grep '\bpista\b'

# Példák II.

- Cat fájl|grep [-a]
  - Cat fájl|grep [a-] # A – vagy az a betű keresése
  - Cat fájl|grep [a-c] # Ugyanaz. Az a és a – keresése
  - Cat fájl|grep [-ac] # Az a-c intervallum, azaz a vagy b vagy c betű
  - Cat fájl|grep [-a-c] # A – vagy a vagy c betű
- Cat fájl|grep [FTC]{})# Olyan minta keresés, ami vagy az egyik betű ÉS utána {}]
- Cat fájl|grep [FTC\]{}) # [] jelen belül nincs speciális jelentés, pl \d vagy .
- Cat fájl|grep []FTC{})# Vagy ] vagy ...

# E(F)Grep – mintaillesztés I.

- egrep – bővített grep -E, létezhetnek különböző implementációk
- fgrep - fixed grep, grep -F, minták sorokban találhatók
  - ali|éva , vagy kapcsolat,
  - Példa: ls | egrep "par|pelda"
  - + Előző minta legalább egyszer
    - Példa: ls | egrep "\<p+f" # szó elején 1 vagy több p, majd f betű.
    - ? Előző minta nulla vagy egyszer ismétlődik
      - Példa: ls|egrep „param\d?” #param, param1, param2,..
  - {n} előző karakter pontosan n szer!
    - Példa: cat almafa|egrep ^[a-b]\w{5}

# E(F)Grep – mintaillesztés II.

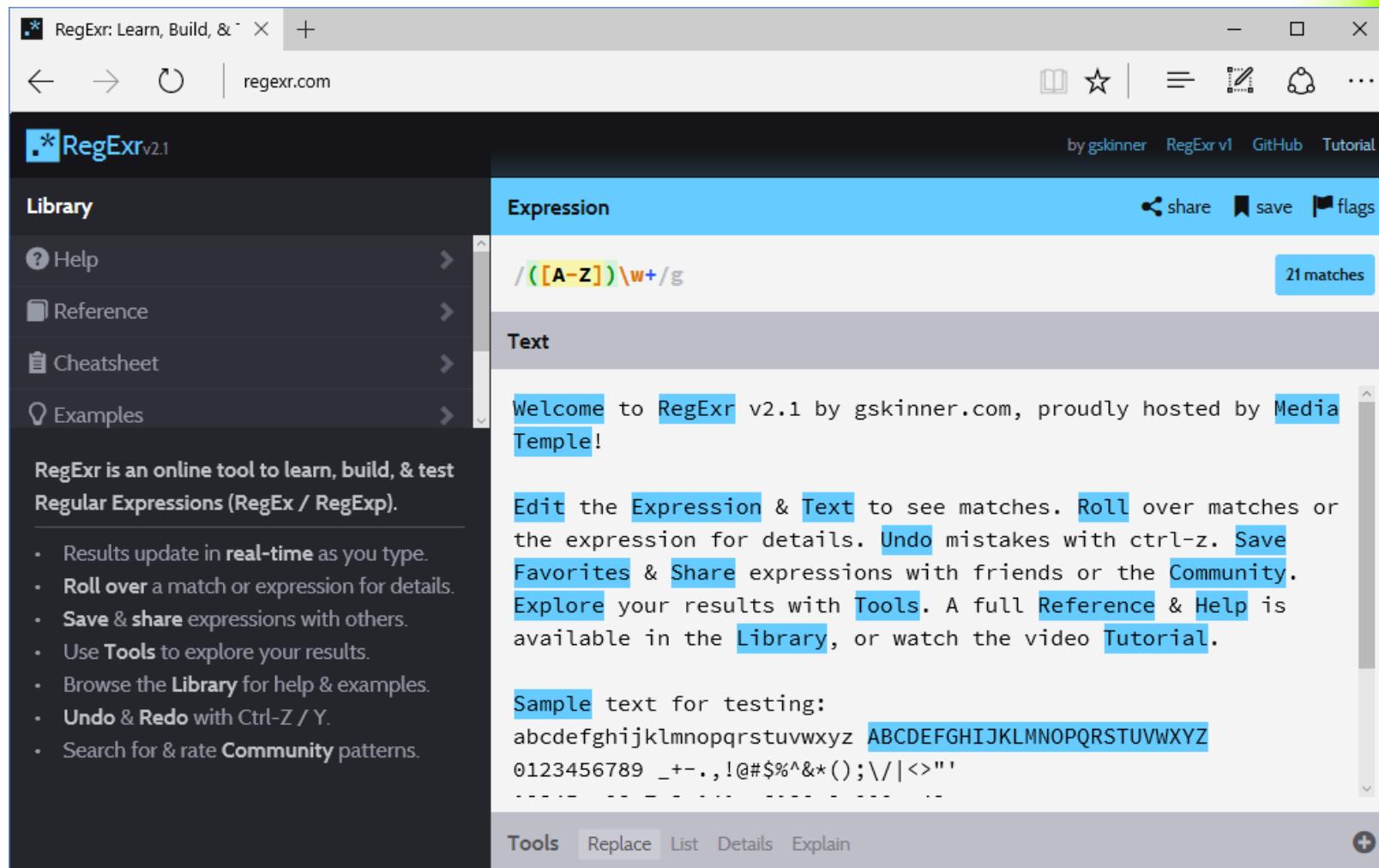
- {2,4} Előző minta 2,3 vagy 4-szer ismételve
  - {1,} Előző minta legalább egyszer
- () Egy csoportba fogunk egy mintát.
  - Ismétlődéshez célszerű, azaz ezt követi egy ismétlésre vonatkozó utasítás +,?,{n}
  - Példa: [0-9]{8}(\s[0-9]{8}){1,2} #bankszámla
    - A [0-9] helyett \d is jó lenne.
- Speciális karaktert célszerű(kell) \ mögé írni.
  - Példa: \.\$ #A sor végén pont van!
  - Példa: ^[+-]?\d+([,.]\d+)? #előjeles szám tizedes ponttal, vagy vesszővel, nem kell \ mert [] belül van!
  - Többi lehetőséghöz: man

# További minták

- E-mail cím(rövidebb változat):
  - \w+[-.\_]?.\*@\w+(\.\w+)+
- Óra:perc:másodperc
  - (([01][0-9])|(2[0-3])):[0-5][0-9]:[0-5][0-9]
- Stb.
- minden programozási nyelv tartalmazza ezt vagy ennek kicsit bővített, módosított lehetőségeit!

# Reguláris kifejezések -online

- Több online rendszer is elérhető
  - <http://regexr.com>
  - <http://regex101.com>
  - Stb.



# Szövegszerkesztők – I.

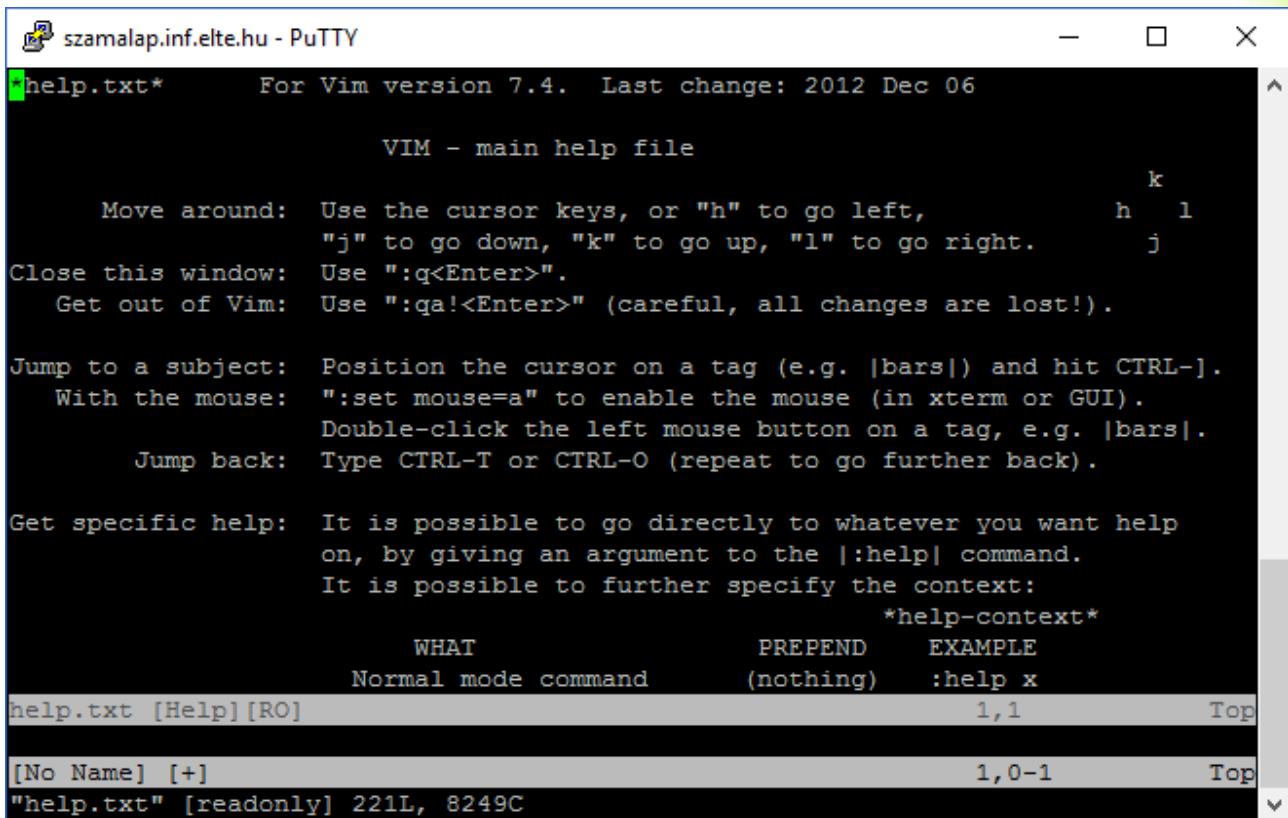
- Széles skála áll rendelkezésre!
- Jellemzően: vi(m), pico, mcedit, joe, stb.
- Alapvető használatukban a segítség a képernyőn!(joe)

Help	Screen	turn off with ^KH	more help with ESC . (^[.)				
<u>CURSOR</u>	<u>GO TO</u>	<u>BLOCK</u>	<u>DELETE</u>	<u>MISC</u>	<u>EXIT</u>		
	^B left	^F right	^U prev. screen	^KB begin	^D char.	^KJ reformat	^KX save
	^P up	^N down	^V next screen	^KK end	^Y line	^KA center	^C abort
	^Z previous word	^A beg. of line	^KM move	^W >word	^T options	^KZ shell	
	^X next word	^E end of line	^KC copy	^O word<	^R refresh	<u>FILE</u>	
	<u>SEARCH</u>	^KU top of file	^KW file	^J >line	<u>SPELL</u>	^KE edit	
		^KF find text	^KV end of file	^KY delete	^_ undo	^N word	^KR insert
		^L find next	^KL to line No.	^K/ filter	^^ redo	^L file	^KD save

IW Unnamed Row 1 Col 1 8:38 Ctrl-K H for help

# Szövegszerkesztők – II.

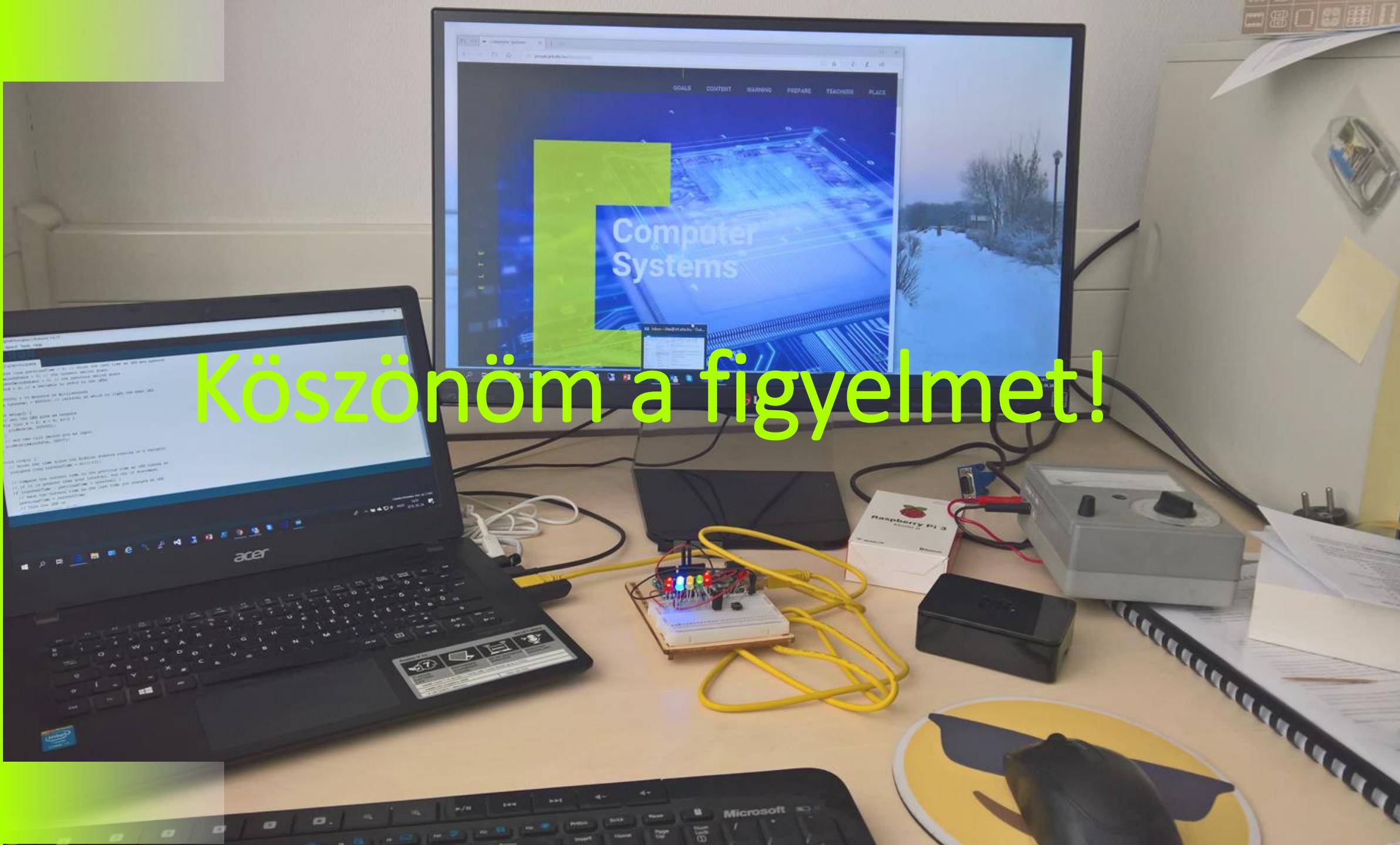
- Néhány vi jellemző!
  - Kétféle üzemmód, parancs vagy szerkesztő mód!
  - Parancsmódból szerkesztő módba: i, a, o, stb
  - Szerkesztő módból parancs módba: ESC
  - Parancs módban: mentés: w név, mentés és kilépés: wq (quit)
  - :help – a képen látható
    - Kilépés help-ből: :q
    - Kilépés mentés nélkül: q!



The screenshot shows a PuTTY terminal window titled "szamalap.inf.elte.hu - PuTTY". The window displays the Vim 7.4 main help file, which is a text document named "help.txt". The text in the window describes various Vim commands and their functions, such as cursor movement (h, j, k, l), closing windows (:q<Enter>), exiting Vim (:qa!<Enter>), jumping to subjects, using the mouse, jumping back, and getting specific help. At the bottom of the screen, there is a command-line interface with several tabs and status bars. One tab is labeled "help.txt [Help] [RO]" and another is "[No Name] [+]".

# VI(M) – névtelen- nevesített buffer

- Cut – Copy – Paste
  - Cut : dd, aktuális sor törlése, névtelen bufferbe helyezése
  - Copy : yy, aktuális sor másolása névtelen bufferbe
  - Paste : p, buffer tartalmának beszúrása
  - Több sor törlése, másolása: 5dd vagy 4yy, öt sor törlése, 4 sor másolása bufferbe.
- Nevesített buffer használata
  - Egybetűs nevet használhatunk, macskaköröm után
  - "s2yy # Az S bufferbe bekerül a 2 aktuális sor
  - "sp # Az s buffer tartalmának beszúrása



Köszönöm a figyelmet!



## 2. Operációs rendszerek feladatai

# Visszatekintés

- Elérhetőség, tárgy teljesítés
- A tárgy célja, tartalma
- Számítógépek tegnap, ma, holnap
- Jelek, információk
- Információk tárolása
  - Fixpontos, lebegőpontos számábrázolás
  - ASCII, UTF-8 stb kódtáblák
- Felépítés, fontosabb elemek
- Operációs rendszerek

# Ami ezután következik...

- Szoftver eszközökre fókuszálunk
  - Gépi kódtól az operációs rendszerig
- Operációs rendszerek és programozási lehetőségei
  - UNIX (Linux) lehetőségek
    - Nyelvi eszközök használata (gépi kód,C++, Java,stb.)
    - Shell script
  - Windows
    - Batch, nyelvi eszközök (gépi kód, C++, Java,stb.)
    - PowerShell

# Mi jön ma?

- Operációs rendszer feladatai
- Operációs rendszer szolgáltatások
- Felhasználói felületek
  - Karakteres, grafikus
- Fájlrendszer, szerepük
- Fájlrendszer jogosultságok
- Alapvető műveletek
  - Fontosabb operációs rendszer parancsok

# Operációs rendszer feladatai

- Megfelelő felhasználói felület biztosítása
- Fájlok kezelése, tárolása
- Perifériák kezelése
- Hálózati szolgáltatások támogatása
- Alapvető feladatok megvalósítása
  - Elemi szövegszerkesztés
  - Hálózat kezelés
  - Stb.

# Operációs rendszer fontosabb szolgáltatások

- Kliens – szerver különbségek
- Közös, osztott háttértár használata
- Közös nyomtatási szolgáltatás használata
- Szervizek kezelése
  - Levelezés, web, terminál elérés stb.
  - Hálózati szolgáltatások (DNS, DHCP, stb)
- Felhasználók kezelése
  - Információs adatbázis

# Unix (Linux) története

- 60-as évek, AT&T Bell Lab, Dennis Ritchie, Ken Thomson
- Egyetemek számára ingyenes
- 80-as évekre a gyártók (HP, IBM, Sun, SGI, DEC, stb) saját termékükre szabják
  - Több változat(HPUX, Solaris, Irix, stb)
  - 2 fő irányzat(AT&T System V, BSD)
- Megszűnik az ingyenesség (kivéve BSD)
- Szabványosítás: POSIX
- 90-es évek, LINUX (Linus Torvalds)

# Karakteres felhasználói felületek

- Linux (putty)
- Windows (CMD)
- Terminál emulátor
  - Karakter beállítás
  - Karakter készlet
  - Terminál típus

The screenshot displays two terminal windows. The top window is a PuTTY session titled 'szamalap.inf.elte.hu - PuTTY' connected to a Linux server (pandora). It shows a welcome message from the Linux distribution and instructions for connecting to an Oracle database via SQLPlus. The bottom window is a Microsoft Windows Command Prompt titled 'Command Prompt' (Version 10.0.10240), showing the user's home directory (C:\Users\illes) and a command to convert files between ISO and UTF-8 formats. Both windows show the output of disk quota commands for the user 'illes' on the 'labhome.inf.elte.hu:/cluster/home' volume.

```
szamalap.inf.elte.hu - PuTTY
Linux pandora 4.0.9-grsec-pandora #1 SMP Fri Jul 31 18:02:34 CEST 2015 i686
* Oracle adatbázis elérése lehetséges sqlplus clienssel a pandora-n.
Használat: sqlplus username@oradb v sqlplus username@ablinux

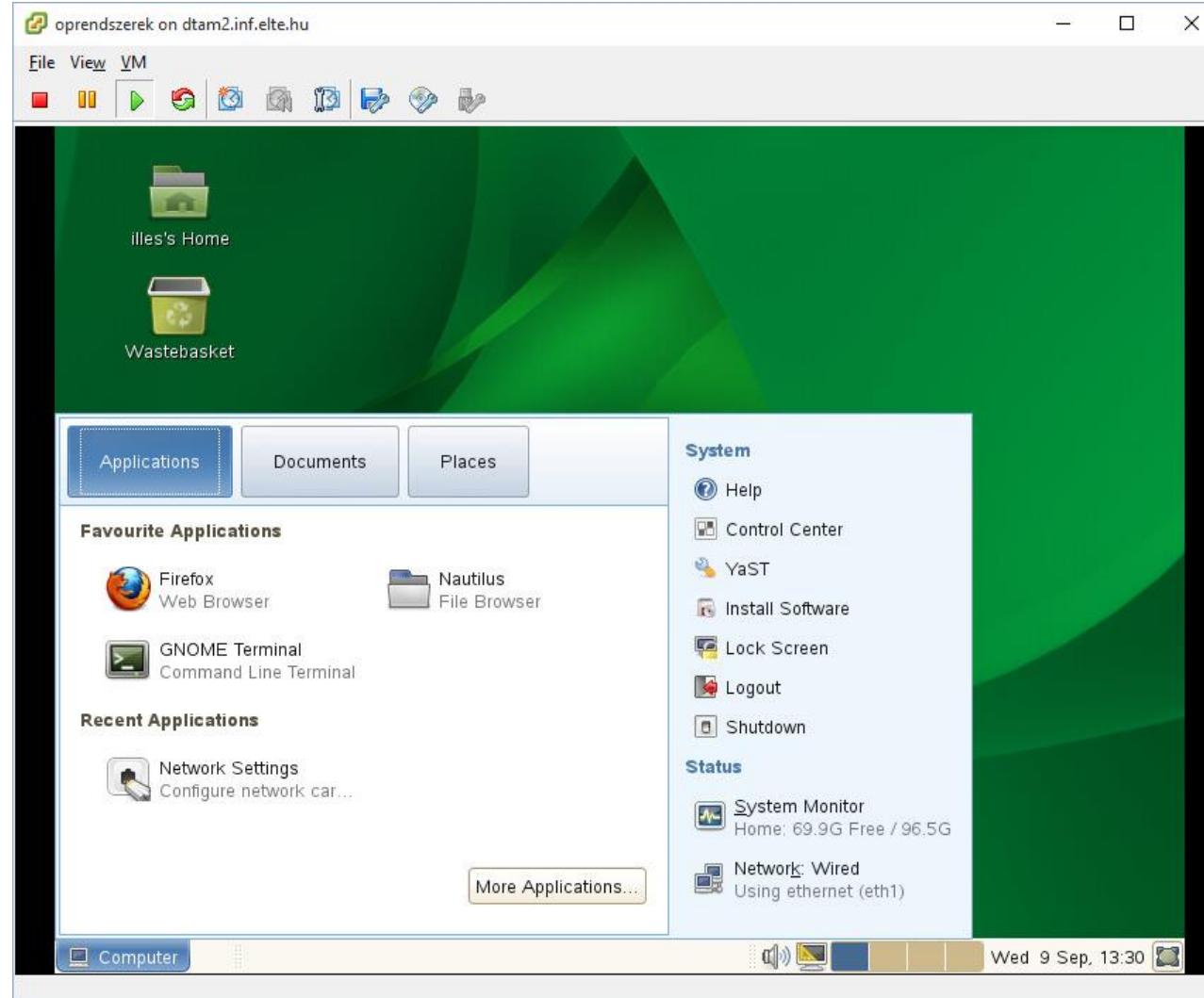
Command Prompt
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\illes>paranccsal lehet konvertálni:
iso.txt >uj_utf8.txt
utf8.txt >uj_iso.txt
JÜÜ
onyvtar elerese laborokbol!

Last login: Sat Sep  5 19:03:39 2015 from 94-21-183-112.pool.digikabel.hu
Disk quotas for user illes (uid 11264):
      Filesystem    blocks   quota   limit   grace   files   quota   limit   grace
labhome.inf.elte.hu:/cluster/home
                                         32  256000  256000          6     0     0
Volume Name           Quota   Used %Used   Partition
user.illes            10485760  1496    0%          0%
Hajra Fradi!
illes@pandora:~$
```

# Grafikus felhasználói felületek

- Grafikus
  - Windows 7,8,10
  - Linux
  - macOS  
(korábban OSX)



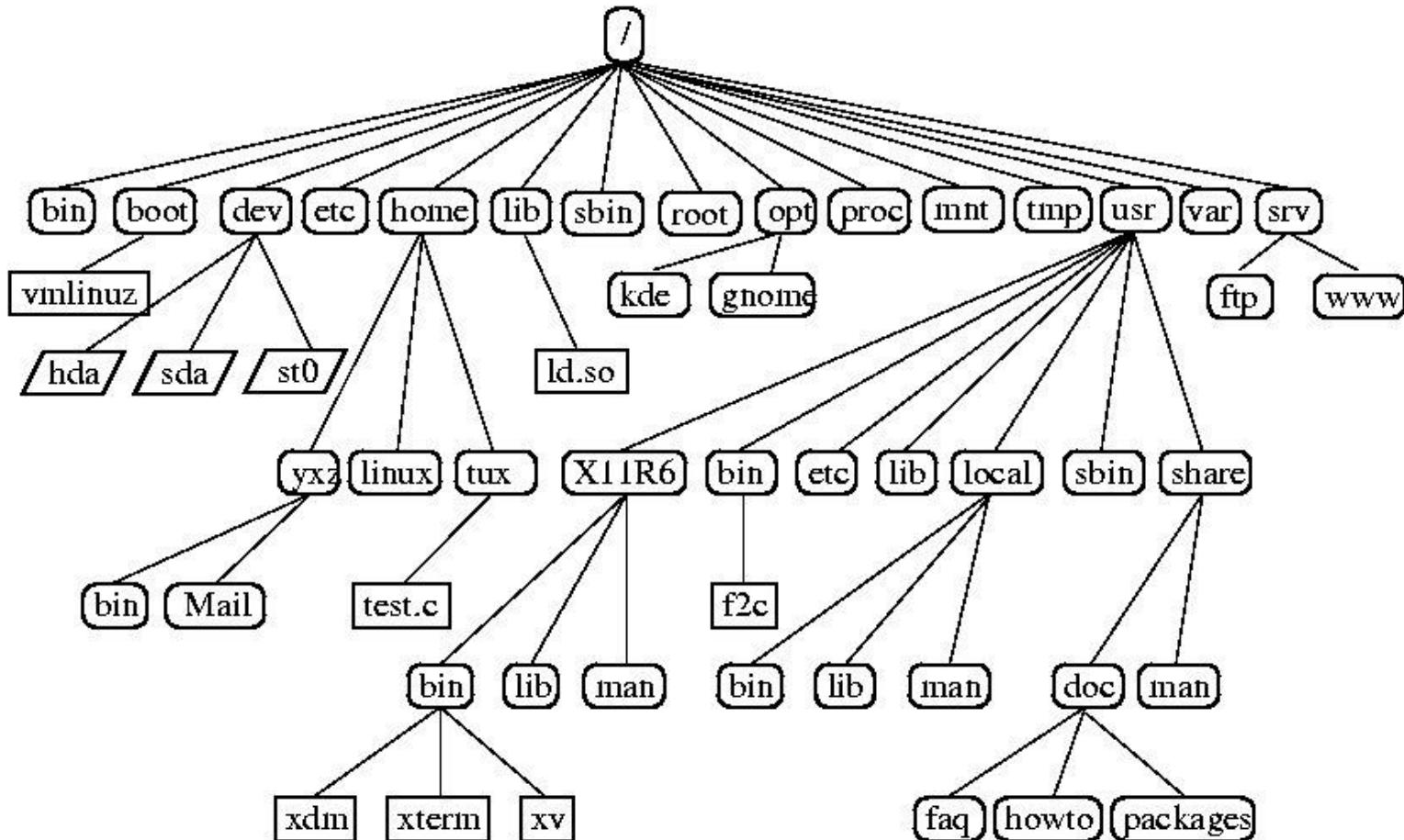
# Népszerű operációs rendszerek

- Ma talán legszélesebb körben használt: Windows
  - A Windows operációs rendszer parancsait, fontosabb lehetőségeit nem tárgyaljuk csak használjuk.
  - Megismerjük a script programok készítésének lehetőségét Windows-ban.
- LINUX – UNIX
  - Egyre népszerűbb, különösen az kutatásban, oktatásban résztvevők körében, de a kiszolgálók zöme LINUX alapú!
  - Megismerjük az alapvető parancsait, lehetőségeit és a script programok készítésének módját.
- macOS (korábban OSX)
- Mobil operációs rendszerek (iOS, Android, WP...)

# Fájlrendszerek, szerepük

- Fastruktúra, Windowsban több belépési pont
- Jellemző Unix könyvtár elemek
  - /, egy gyökér van, ez a /
  - /dev/... az eszközök közös könyvtára
    - pl: /dev/fd0h1440, 1.4 floppy, /dev/null: szemétkosár
  - /etc/... konfigurációs állományok könyvtára
    - pl: /etc/passwd, felhasználók felsorolása
  - /home, /h, felhasználók könyvtára
    - pl:/h/i/illeg
  - /usr/.../usr/local/..., rendszer(helyi )könyvtárak
    - pl: /usr/bin/cc, /bin/sh <-> /usr/bin/sh
  - /var/..., működési segéd, pl. logok

# Open Suse fájlrendszer



# Fontosabb shell típusok

- Shell: klasszikus felhasználói felület programja, a Unix rendszerből származik
- Windowsban is van: CMD
- Unix alatt több
  - Sh (Bourne shell)
  - Ksh (Korn shell)
  - Csh (C shell)
  - Sh (Posix shell, a korábbi Bourne néven)
  - Bash (Bourne again shell)
    - minden felhasználó alapértelmezett shellje!
    - Parancs history, sorszerkesztés, fájlnév befejezés, alias kezelés

# Bash fontosabb jellemzők I

- Fő kapcsolódási pont(mindent ebben végzünk)
- Parancssor szerkesztés, kiegészítés(tab)
  - Ha nem egyértelmű kiírja a választékot.
- Előző parancs(ok) használata (fel-, lenyíl)
  - history n (az előző n parancs kiírása)
- Álnév használat
  - Alias név=szöveg
  - PL: alias dir="ls -l"
  - dir a\*

# Bash fontosabb jellemzők II

- Parancs szerkezet
  - Elsődleges, másodlagos prompt: PS1,PS2
  - Parancs alakja: PS1 név paraméter(ek) (enter)
  - Ha úgy érzi nincs vége a parancsnak, kapjuk a másodlagos promptot!
  - Egy sorba két (több) utasítás: ;
  - Megjegyzés: #
- Login folyamat: /etc/profile, majd ~/.profile végrehajtása
  - Helyi utasítások gyűjtőhelye: .profile, pl: PATH
    - A .profile helyett lehet .bash\_profile vagy .bash\_login is!
  - Kilépés: .bash\_logout

# UNIX fájlrendszer tulajdonságok

- Szerkezete hierarchikus
- Alapvetően 2 féle bejegyzés lehet
  - Könyvtár (jele: d)
  - Fájl (jele: -, pipe jele: p)
- Az eszközök is „fájlnevet” kapnak (/dev könyvtár)
- Link, speciális fájlbejegyzés
  - Hard link, csak fájlra, fájlrendszeren belül, a fájlbejegyzés hivatkozási szám változik (Windowsban mklink)
  - Soft link( jele: l), hasonló mint a „shortcut” Windowsban! (mklink –s)
- Mai változatuk naplózottak, nagyobb biztonság, konzisztencia
  - Ext2, Ext3, Ext4FS, BTRFS(B-tree file system)

# Fájl, könyvtárnevek, konvenciók

- Név hossza nem korlátos!
- Tetszőleges karakter használható!
  - Nem tanácsos használni!
- Javaslat: Ne használunk nevekben helyközt, ékezetes karaktereket, speciális(\*,%,\$,stb) karaktereket!
- Nincs kiterjesztés mint a Windows értelmezésében!
  - .exe, .txt fájlvég létezhet, de semmit nem jelent!
- Ha a kezdőkarakter a . (pont), akkor takart állományt hozunk létre!

# Speciális fájlnév hivatkozások

- Láttuk, ajánlott karakterek fájlnévben: betűk, számok, ., \_, -
- Hogy hivatkozhatunk egyszerre több névre?
- Speciális karakterek: \*, ?, [], !
  - ? egyetlen karakter helyettesítés
  - \* tetszőleges karakter helyettesítés (0 is)
    - \* nem helyettesíti a fájl elején álló pontot!
  - [abc] a felsorolt jelek közül egy
  - [!abc] nem a felsorolt karakterek közül egy
    - [A-Z] nagybetű
    - [1-9] 1,9 közti szám

# Fájl jellemzők

- Név
- Méret
- Létrehozás dátuma
- Tulajdonos
- Tulajdonos csoportja
- Hard link szám
- Jogosítványok

Összesen 31										
drwxr-xr-x	2	illes	10715	2048	dec	13	2013	Asztal		
drwxr-xr-x	2	illes	10715	2048	dec	13	2013	Dokumentumok		
-rw-r--r--	1	illes	10715	19	szept	15	2014	elso		
-rw-r--r--	1	illes	10715	53	okt	2	2014	joetext1		

# Parancsriadás, paraméterek

- Parancs: egy karakterszor az enter-ig!
- Parancsfeldolgozás
  - Az értelmező szétszedi határoló karakterek szerint(helyköz)
  - Első szó: parancs neve
  - Többi szó: paraméterek
  - Grafikus felületen a megfelelő(klikk) esemény elindít egy parancsot!

# Alapvető parancsok I.

- ls, ls -l, ls -al #könyvtár tartalom
- pwd, cd, mkdir, rmdir #könyvtár műveletek
- chmod, chown, chgrp, umask # jogosultság
- passwd # jelszó állítás
- cp, mv, rm, ln # fájl műveletek
  - ln -s #soft link
- mail, telnet(ssh), ftp, nfs(mount) #arpanet
  - ssh név@host
- echo Hajrá Fradi! # Képernyőre (std output) írás
- man ls # ls parancs manuál, help!

# Alapvető parancsok II.

- who, whoami #ki van bejelentkezve
- talk usernév [terminál] # beszélgetés kezdeményezés
  - write user [tty]
- mesg no # kezdeményezés tiltása
- clear – karakteres képernyő törlése
- date #dátum, idő kiírása
  - touch filename # ha nem létezik létrejön különben idő módosul.
- finger user #felhasználói információ kiírása

# Alapvető parancsok III.

- find – keresés
  - find . –name alma.fa
- tar (tape archive) –kulcs [f file] fájlok
  - fontosabb kulcsok:
    - c, create, archive létrehozás
    - x, eXtract, kivesz, visszatölts archivumból
    - t, tartalom kiírása
    - v, képernyőre írja a file neveket
  - Példa: tar –cvf alma.tar \*.txt
    - tar –xvf alma.tar \*

# Alapvető parancsok IV.

- touch fájlnév #módosítás idő állítás, vagy üres fájl
  - Fájl tartalom módosítást lásd szövegszerkesztés, output átirányításnál.
- cat, head, tail # fájl tartalom megnézés
- more, lapozás előre, less lapozás előre, hátra
- read a # billentyűzetről az a nevű változóba olvas be enterig
  - read a b # a és b változóba olvas be, a-ba az első helyközig olvas, majd a többi elem b-be kerül
  - line – a bemenet egy sorát a kimenetre írja
- diff file1 file2 # fájlok összehasonlítása
- zip, unzip, gzip, tömörítés
  - zip alma.zip \*.txt # alma.zip-be tömöríti az összes txt kiterjesztésű fájlt.
- ...és még sok minden....
  - Segít a MAN!

# Hozzáférési jogosultságok I.

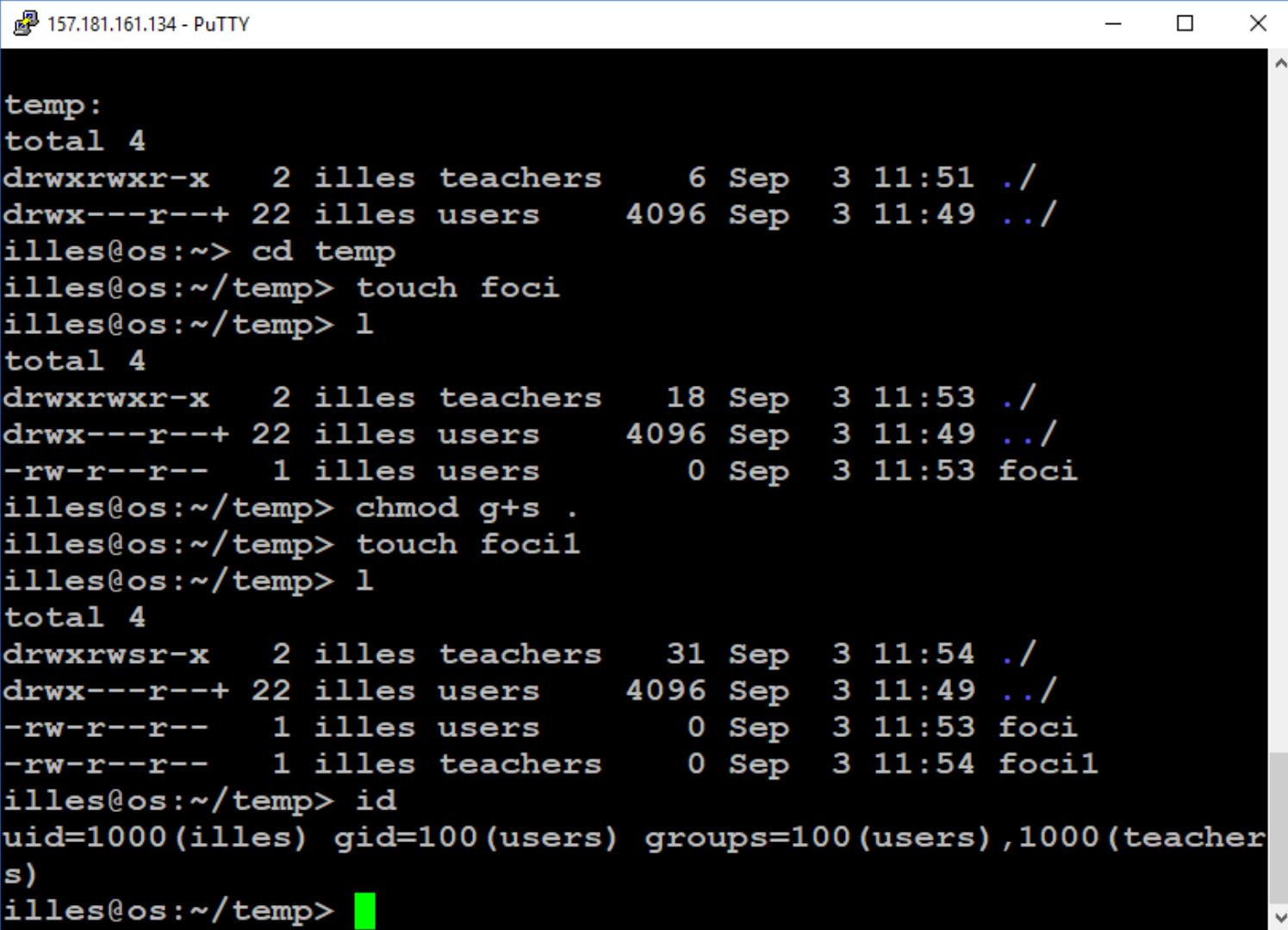
- Alapvetően egy 3x3-as rendszer él (oktális rendszer)
  - minden bejegyzésnek van
    - Tulajdonosi jogosultsága (u)
    - Csoport jogosultság (g)
    - mindenki más(others) jogosultsága(o)
  - minden jogosultság három részből áll
    - R – olvasási jog
    - W – írási jog
    - X – végrehajtási jog
- Jog állítás: chmod g+w alma.fa

# Hozzáférési jogosultságok II.

- Kezelhetjük a r,w,x jogosítványokat mint 3 bites számot!
  - 8-as számrendszer!
- Alapértelmezett jogosultság: 644
- umask, azon bitek megadása, melyekhez nem adunk jogot
  - példa: umask 111 # az új file rw-rw-rw- jogú
  - default: umask 022
- Kiegészítő jogok: Példa: chmod 6644 alma
  - setuid, parancs a fájl jogaival fut, nem a futtató jogaival (x helyett S )
  - setgid, parancs a fájl csoport jogaival fut
  - sticky bit, fájl, könyvtárban csak saját fájl törölhető

# Kiegészítő jogok – Csoport öröklése

- GUID bit könyvtáron:  
a könyvtárban létrejövő  
új fájl nem az  
elsődleges hanem az  
aktuális könyvtár  
csoportjának  
tulajdonába kerül!
- Fontos: Az umask él!  
(Ha akarok írás jogot, az  
umaskban is meg kell  
adnom!)



The screenshot shows a PuTTY terminal window titled "157.181.161.134 - PuTTY". The terminal displays a series of Linux commands and their outputs:

```
temp:  
total 4  
drwxrwxr-x 2 illes teachers 6 Sep 3 11:51 ./  
drwx---r---+ 22 illes users 4096 Sep 3 11:49 ../  
illes@os:~/temp> cd temp  
illes@os:~/temp> touch foci  
illes@os:~/temp> l  
total 4  
drwxrwxr-x 2 illes teachers 18 Sep 3 11:53 ./  
drwx---r---+ 22 illes users 4096 Sep 3 11:49 ../  
-rw-r--r-- 1 illes users 0 Sep 3 11:53 foci  
illes@os:~/temp> chmod g+s .  
illes@os:~/temp> touch foci1  
illes@os:~/temp> l  
total 4  
drwxrwsr-x 2 illes teachers 31 Sep 3 11:54 ./  
drwx---r---+ 22 illes users 4096 Sep 3 11:49 ../  
-rw-r--r-- 1 illes users 0 Sep 3 11:53 foci  
-rw-r--r-- 1 illes teachers 0 Sep 3 11:54 foci1  
illes@os:~/temp> id  
uid=1000(illes) gid=100(users) groups=100(users),1000(teachers)  
illes@os:~/temp>
```

# Hozzáférési jogosultságok III.

- Ez a jogosultság állítás egész jó, de nem az igazi!
  - Például Windows alatt minden állományhoz egyenként adhatunk felhasználókat különböző jogokkal!
  - Hogy lehet ezt Unix/Linux alatt megcsinálni?
- Megoldás: ACL (Access Control List)
  - setfacl – beállítás
    - setfacl -R -m u:Pityu:rwx alma.fa # rekurzívan, modify
    - setfacl -d -m u:Pityu:rwx konyvtar # konyvtar-ban keletkező új fájlok öröklik # a default) jogokat
    - setfacl -b alma.fa # ACL jogok törlése
    - Bővebben: man
  - getfacl – beolvasás
    - getfacl alma.fa

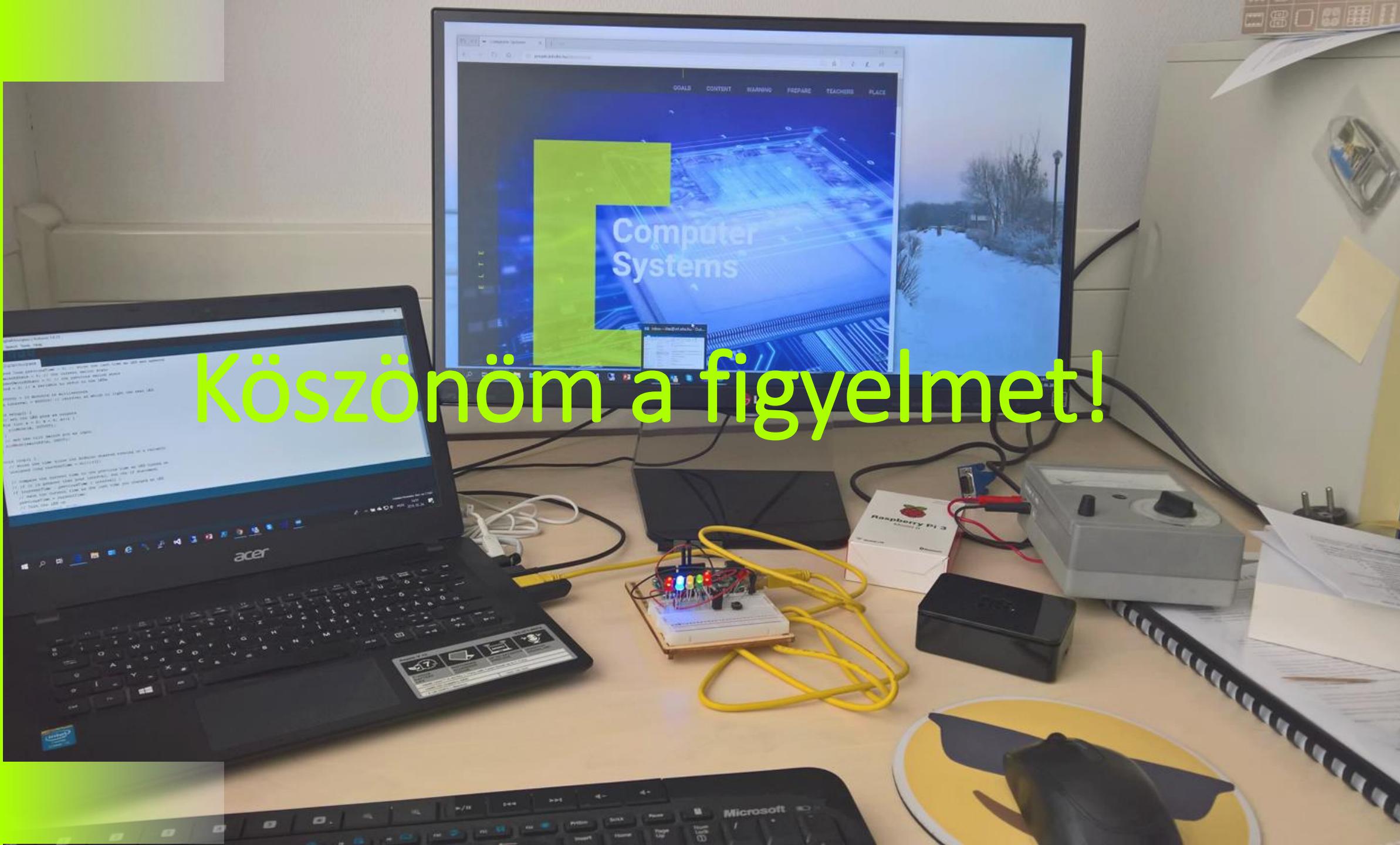
# Jogosultságok-Fájlrendszerek

- A korábbi jogosultság állítás lehetőségek klasszikus Linux/Unix fájlrendszereken használható. (Ext2FS, Ext3FS, BTRFS, stb.)
  - df – display filesystems, a becsatolt fájlrendszerek listázása
- Elosztott fájlrendszerek esetében ez módosulhat!
  - NFS (Network File System) esetében használhatók a korábbiak.
  - AFS (Andrew File System) esetén más, könyvtárszintű parancsok.

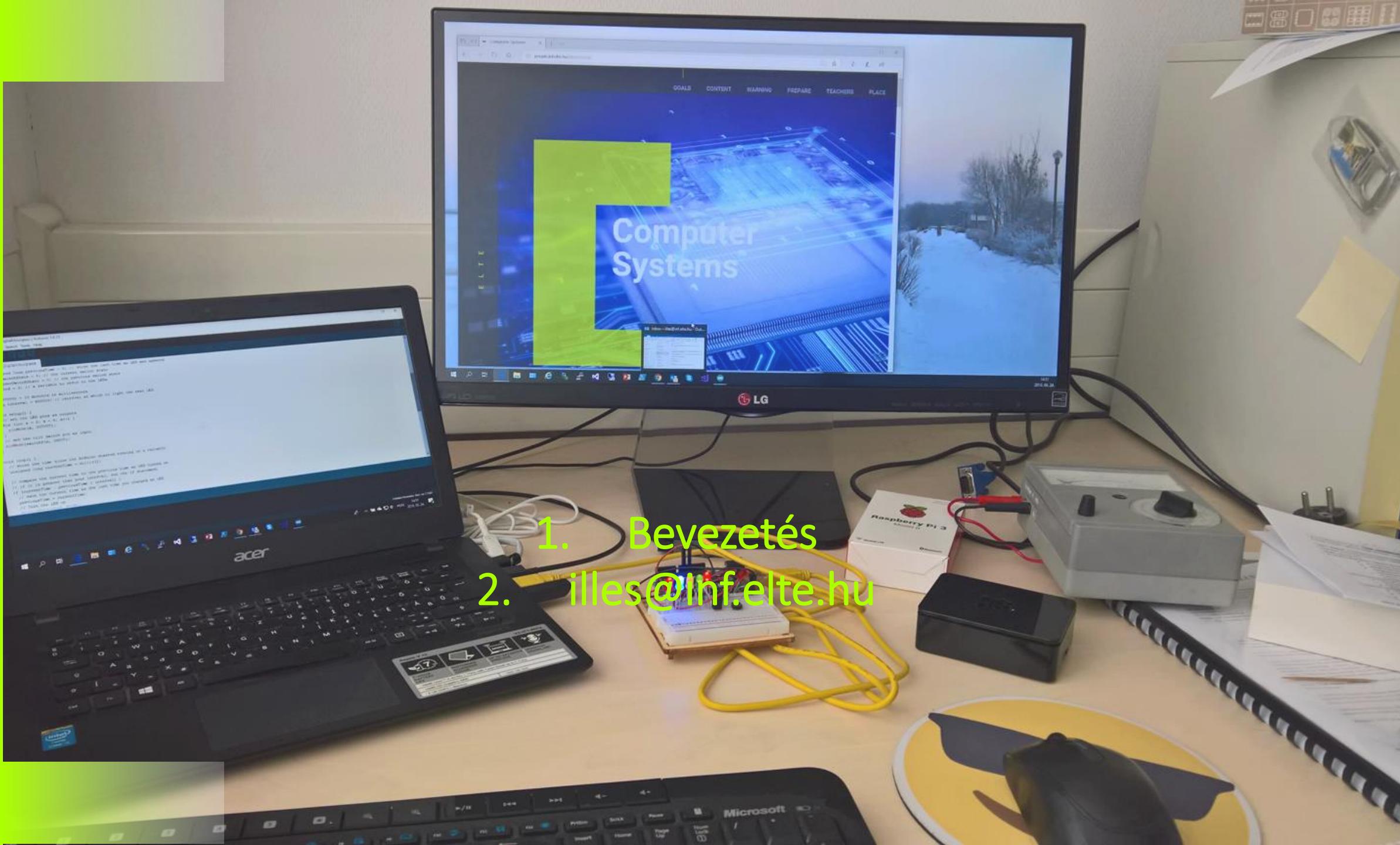
```
AFS                               2147483647      0 2147483647  0% /afs
illes@valerie:~$ cd ..
illes@valerie:/afs/inf.elte.hu/user/i/il$ █
```

# AFS jogosítvány állítás

- fs parancs a fő adminisztratív lehetőség az AFS (OpenAFS) fájlrendszerhez
- Jogosítványok listázása: fs listacl könyvtárnev (fs la)(la – listacl rövidítése)
- Jogosítványok állítása: fs setacl könyvtárnev (fs sa)
- Az fs parancs egyik funkciója a jogosítványok kezelése! A teljes leíráshoz: man fs
  - man fs\_listacl vagy man fs\_setacl
  - Vagy: [http://docs.openafs.org/Reference/1/fs\\_setacl.html](http://docs.openafs.org/Reference/1/fs_setacl.html)



Köszönöm a figyelmet!



# A tantárgy célja

- A számítógépek szerepének, feladatainak megismerése
- Adjon olyan alapismeretet ami a további tantárgyakhoz szükséges
- Próbáljon egy közös nevezőt adni a sokféle előismerettel érkező hallgatóknak
- Számítógépek felépítését, elemeit, alap működését mutassa be
- Unix, Windows rendszerek alapvető parancsait ismertesse meg
- Sajátítsák el, ismerjék meg az alapvető script programozás lehetőségeit
  - Unix (Linux) shell script, Windows Powershell

# A tantárgy tartalma

- Információ robbanás, számítógépek tegnap, ma, holnap
- Architektúrák, fontosabb elemek
- Operációs rendszerek szerepe
- A Linux, Windows rendszerek alapvető lehetőségei
  - Parancsok, hálózati lehetőségek, alapvető alkalmazások
- Programozni, programozni, programozni...
- Unix shell script
- Powershell

# Elérhetőség, információ

- A „Számítógépes rendszerek” tárgy honlapja: <http://szamrend.inf.elte.hu>
- A tárgy órabeosztása: 2+2+1(esti 1+1)
- A tárgy kreditértéke: 5
- Teljesítés eredménye: összevont jegy (X-es tárgy)
- Az összevont jegy (gyakorlati jegy) feltételei:
  - 4 zárthelyi dolgozat, minden dolgozat eredmény  $\geq 2$  (50%)
    - Gyakorlaton lesznek a ZH-k, utolsó alkalommal 2 (PS+előadás)
  - 3 beadandó feladat (határidő betartás)
  - Félév végén pótzs lehetőség!

# Irodalomjegyzék

- Támop online tananyag.
  - <http://www.tankonyvtar.hu/>
- Brian W.Kernighan, Rob Pike: A Unix operációs rendszer
- Unix manual (man)
- <https://www.linux.com/learn>
- <http://www.microsoftvirtualacademy.com/>
- <http://www.powershell.com>
  - <http://mek.oszk.hu/10400/10402/> (Magyar nyelven: PS 2.0 leírás)

# Miről beszélünk ma?

- Mi a tantárgy célja, miért jó ez a tárgy?
- Számítógépek tegnap-ma-holnap
- Jelek, információk, tárolásuk
- Számok, karakterek tárolása, kódolás
- Számítógépek felépítése, fontosabb elemek
- Hardver-Szoftver
- Operációs rendszer szerepe
- Alapvető lépések

# Számítógépes rendszerek, fogalmak

- Általában a tantárgyi fogalmak, témaköörök, anyagrészek leírásáról
  - Törekszünk szemléletes, egyszerű, érthető, tömör megfogalmazást használni!
  - Nem törekszünk a teljesség igényével megfogalmazott részletességre!- Nagy témaör, az informatika utóbbi 50 évének legfontosabb elemei ehhez tartoznak!
- Számológép – Számítógép fogalma
  - Számológép, számítógép „elődje”, egyszerű, napi matematikai számolások segítése, hétköznapokban megjelent kb. 40 éve.
  - Számítógép, az egyszerű számolásokon túl, általános számítási, vezérlési stb feladatok elvégzésére.
- Általános számítógép – Célszámítógép
  - Általános – Operációs rendszer megjelenés
  - Célszámítógép – Folyamat szabályozások, intelligens eszközök.

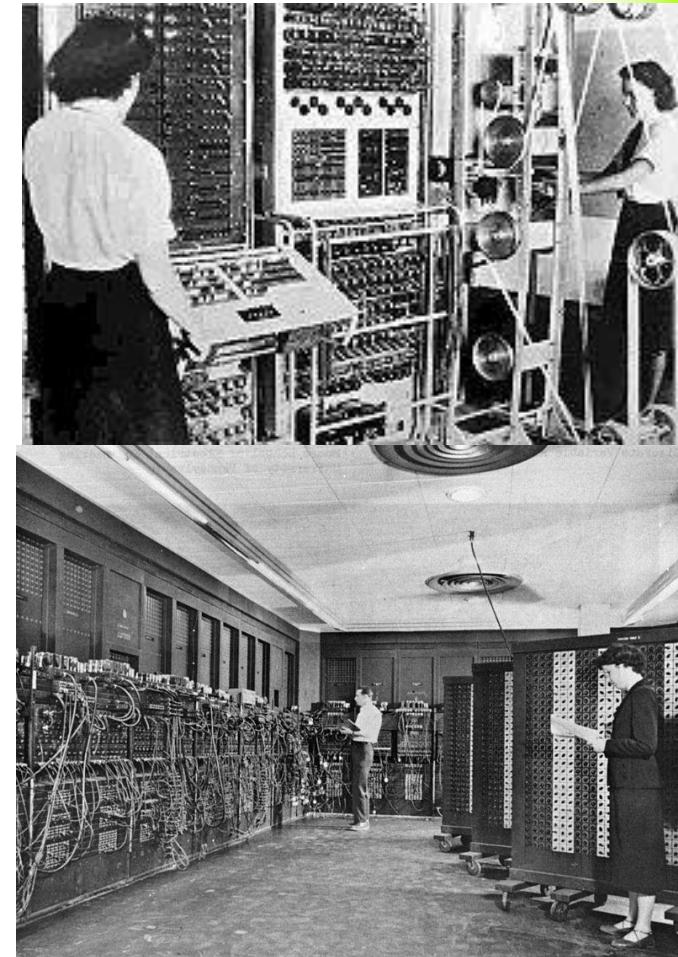
# Információs robbanás a világban

A mai generáció  
így születik!



# Számítógépek tegnap I.

- Ehhez a periódushoz értem a kb. 1980-as évekig tartó időszakot.
  - Nem azonos a klasszikus generációs felosztással!
- Jórészt „számológépi” feladatok!
- Jellemző kulcselemek:
  - Abakusz (szcsotka), mechanikus, elektromechanikus gépek
  - 1943: Alan Turing Colossus gépe
  - 1946: ENIAC (*Electronic Numerical Integrator And Computer*), 10-es számrendszer! 30tonna!



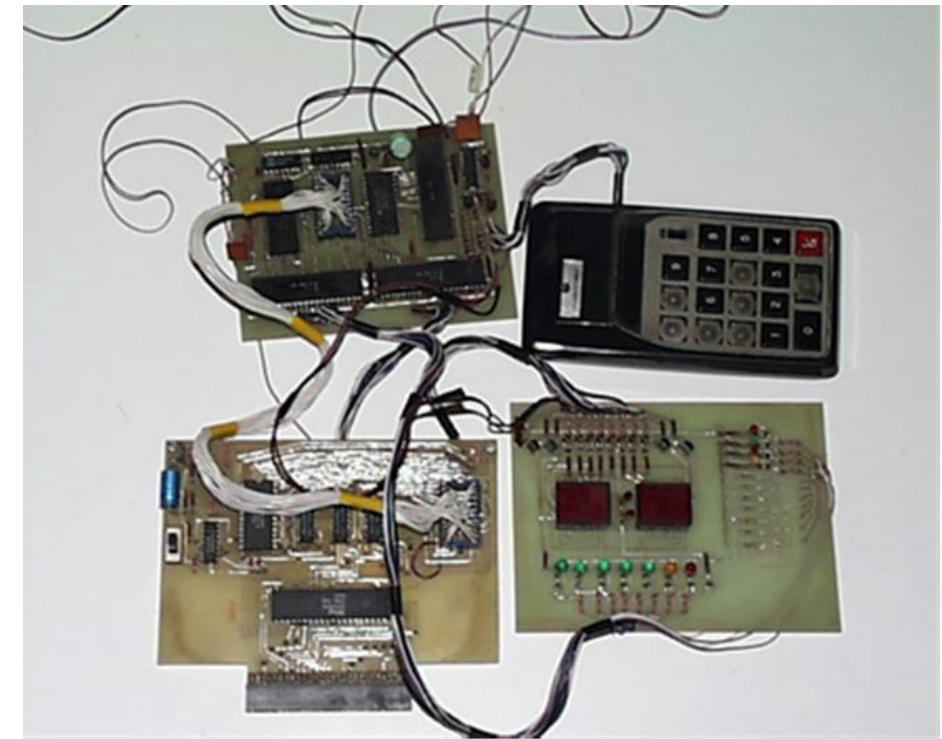
# Számítógépek tegnap II.

- 1949 – EDVAC (*Electronic Discrete Variable Automatic Computer*) - Kettes számrendszer, digitális elv
- 1964: IBM System/360
- Hazai vonatkozás: DEC PDP11/40- KFKI TPA 1140
  - Soros terminálok, Fortran fordító, modularitás, „szolid” méret!
- Kialakul az operációs rendszer!



# Számítógépek tegnap III.

- Célszámítógépek léteztek kicsiben is!
  - Mai IoT rendszerek, „board”-ok elődje.
  - A képen látható prototípus „IoT” Z80 alapú.
  - ZX\_Spectrum a fejlesztő számítógép.
  - 1980-as évek
  - 90-es években vált általánossá (PIC vezérlők)



# Számítógépek ma I.

- Folytatódik az elektronikai eszközök méretcsökkenése, teljesítmény, kapacitás növekedése!
  - Kisebb, nagyobb teljesítményű processzorok, háttértárak.
- Operációs rendszerek fejlődése, virtualizáció!
- Egy számítógép nem számítógép! Hálózatok!
  - „Felhő” szolgáltatások, univerzális információ elérés!
- Elindul a „számítógép vezérelte eszközök” térhódítása!
  - Kezdődött talán a telefonnal(okostelefon), ki tudja hol áll meg!
  - Okos eszközök, IoT (Internet of Things)

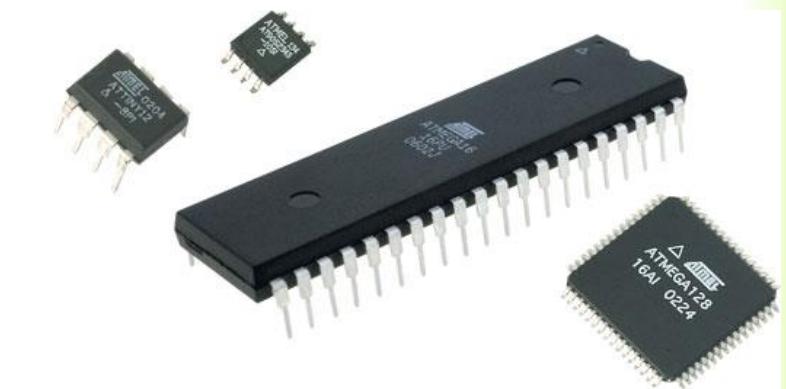
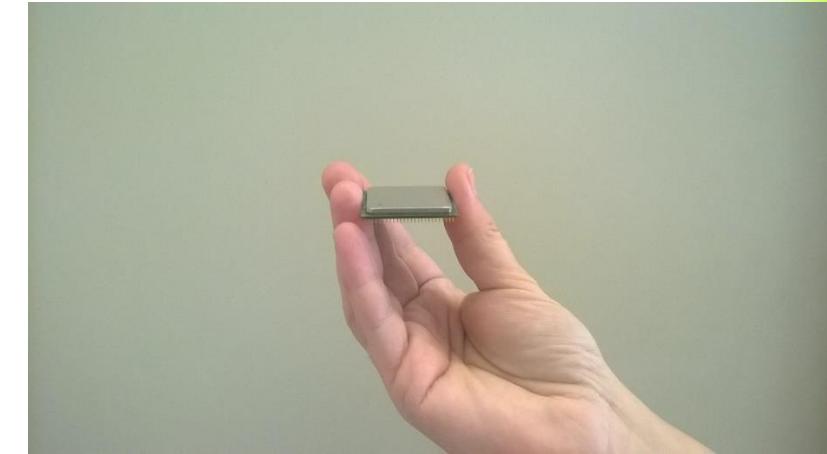
# Számítógépek ma II.

- Jellemző mai adatok.
  - Processzor típus, hány darab van benne
  - Memória méret
  - Háttértár
- 1 processzor, több(4,6,8,10,12) mag
- HPC (High Performance Computing)
  - Debrecen- HPC: (SGI)1536 mag, 165. volt a ranglistán
    - MIPS,FLOPS, <http://www.top500.org> – aktuális lista
  - ELTE-atlasz: 90 darab 4 magos processzor(1 fejgép+44 node)



# Számítógépek ma III.

- Mikroprocesszorok – Mikrokontrollerek
  - Milyen gyors? MHz, GHz
  - CISC-RISC
  - Hány bites?
    - Mai mikroprocesszorok gyakorlatig mind 64 bitesek.
    - A mikrokontrollerek viszont jellemzően 8 bitesek!
  - Cache szerepe a mikroprocesszorban!
  - TLB szerepe a mikroprocesszorban!
  - Neumann architektúra
  - Harvard architektúra
    - (adat, utasításmemória külön)



# Mi kell egy számítógéphez?

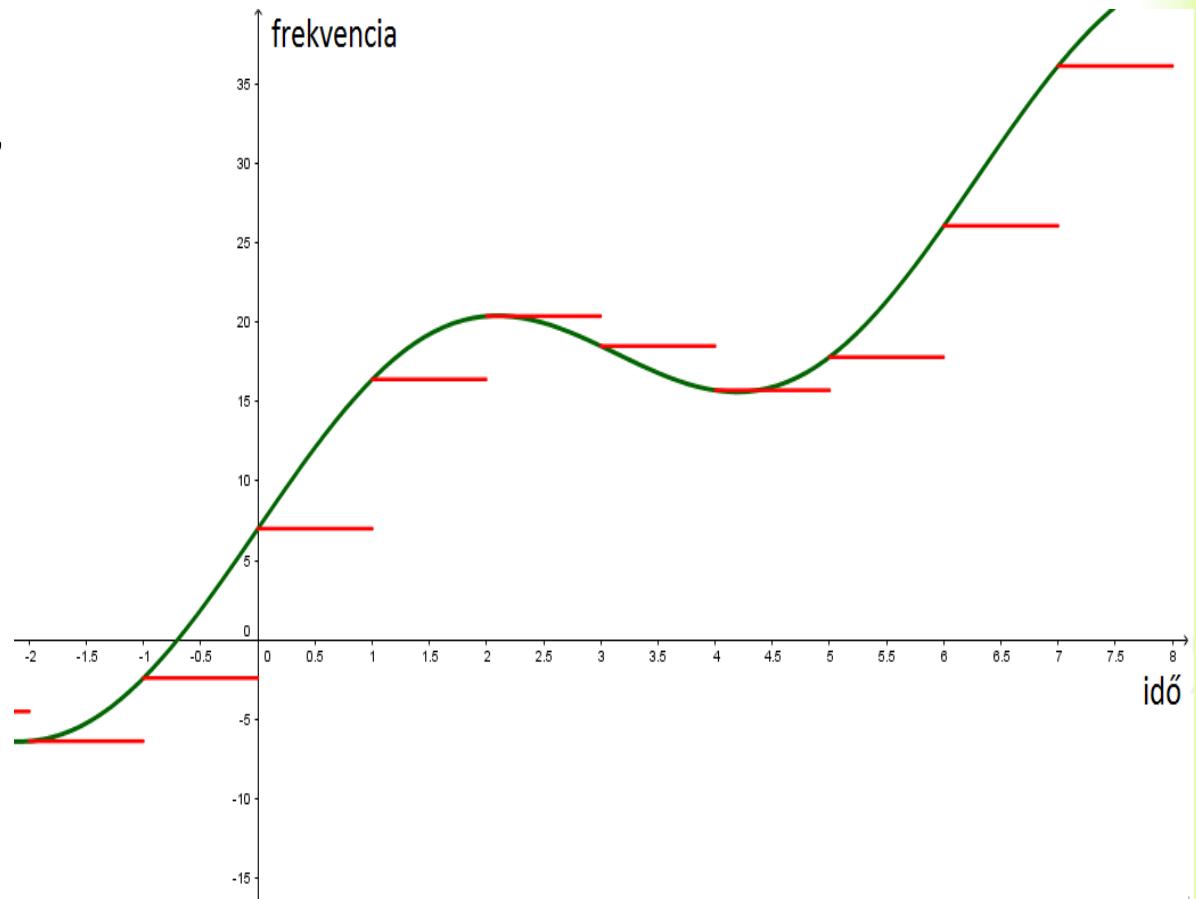
- Kötelező elemek:
  - CPU – a bekapcsolás után automatikusan elkezdi végrehajtani az utasításokat.
  - Memória – ebben tároljuk az utasításokat a processzor számára.
    - Kétféle típus alapvetően – ROM,RAM
- Választható elemek:
  - Háttértár – jellemzően merevlemez
  - Periféria egységek – ezek biztosítják a számítógép kapcsolatát a külvilággal.

# Számítógépek holnap

- Már ma is érzékelhető, a számítógépek egyre több tevékenységet átvesznek az embertől, az automatizáció folytatódik!
- Kapacitások növekedése, a mennyiségi növekedés minőségi változásokat hoz, mesterséges intelligencia erősödése!
- Vizuális információk feldolgozásának erősödése, verbális kommunikáció kialakulása!
- Erősödik a „SZOFTVER” szerepe! Mesterséges intelligencia!
- Hódít a számítógép...”ki tudja hol áll meg, kit hogyan talál meg..” (Arany)

# Jelek, információk

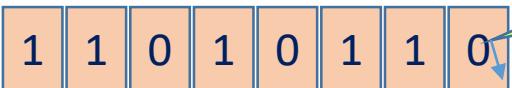
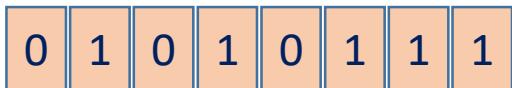
- Analóg jel, információ, folytonos jelértékek! A környezeti paraméterek, távolság, hőmérséklet, zene, zaj, áramerősség stb. természetes értékei!
- Digitális jel, információ, diszkrét, nem folytonos értékek tárolása! Pl. CD-n tárolt zene.



# Információ (jel) tárolása

- Bár voltak analóg számítógépek, de gyakorlatilag az 50-es évektől a digitális elv él.
- Hogy tároljuk a diszkrét értékeket?
  - Volt 10-es számrendszerbeli ábrázolás is (1946,ENIAC).
  - Azóta gyakorlatilag a **2-es (bináris)** számrendszer a meghatározó!
    - A 8-as, 16-os szintén gyakran megjelenik, de csak a könnyebb leírás miatt!
    - Bit, Byte, Word,Kilobyte(KB), Megabyte(MB), Gigabyte(GB), Terrabyte(TB), (peta,exa,zetta,yotta)
      - $1024 (2^{10})$  a váltószám ( $1KB=1024\text{byte}$ ,  $1MB=1024KB$ ), kivéve bit-byte (8)

Word(szó)



Bit

Byte

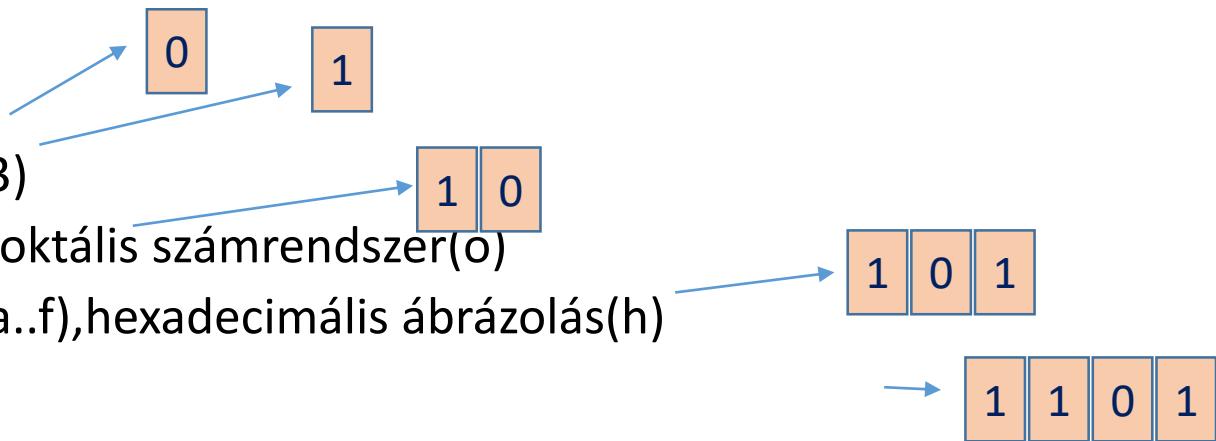
Számítógépes rendszerek

# Számok ábrázolása I.

- Alap: bináris ábrázolás
- Fixpontos ábrázolás-fix számú bit

- Természetes számok (N) :

- 1 biten 2 különböző érték (0,1)
    - 2 biten 4 különböző érték (0,1,2,3)
    - 3 biten 8 különböző érték (0,..7), oktális számrendszer(o)
    - 4 biten 16 különböző érték(0,..9,a..f),hexadecimális ábrázolás(h)
    - 8 bit -> 1 byte (0-255)
    - 16 bit, 2 byte (0-65535,  $2^{16}-1$ )
    - Hány bitet használhatunk egy természetes szám ábrázolására?



# Számok ábrázolása II.

- Egész számok ábrázolása – jellemzően 4 byte-on tároljuk
  - Egyes komplementensű ábrázolás: első bit legyen az előjel!
    - Negálás: egy bit „ellenkezője” annak negáltja ( $0 \rightarrow 1$ ,  $1 \rightarrow 0$ )
    - Egy szám negatív változatát úgy kapjuk ha negáljuk a biteket!
      - $-x = \text{negált } x$
      - Jellemzője: 2 nulla van! 😊
      - 1 bájton így  $-127 + 127$  közötti számok ábrázolhatók!
  - Kettes komplementensű ábrázolás
    - $-x = \text{negált } x + 1$
    - Egy nulla,  $-128 + 127$  közti számok egy bájton.

Decimális szám	Bináris számábrázolások		
	Előjel és abszolút érték	Egyes komplementens	Kettes komplementens
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000 1000	0000 1111	0000
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001
-8	-	-	1000

# Számok ábrázolása III.

- Mi a helyzet a tört számokkal? Például: 3,14159265358979
  - Nem probléma! Ábrázoljuk az egészrészt illetve a törtrészt egymás után!
  - Nem az igazi megoldás, ugyanis nagy a helyigénye!
- Lebegőpontos számábrázolás.
  - Legyen a szám normál alakja:  $+/- M * A^K$ ,  $M < 1$ 
    - M- mantissa, A-hatvány alap, K- hatvánnyitevő(karakterisztika)
      - Példa: A legyen 10, akkor az 517 alakja:  $0,517 * 10^3$
      - Példa: A=2 esetén,  $517 = 1000000101 \rightarrow 0,1000000101 * 2^{10}$
    - 4 bájtos ábrázolás: 1 bit->előjel, 8 bit->karakterisztika, 23 bit mantissa(IEEE754)
    - 8 bájtos ábrázolás: 1 bit->előjel, 11 bit->karakterisztika, 52 bit mantissa(IEEE754)
      - Mekkora a legnagyobb ábrázolható szám?

# Kódolás, karakterek tárolása

- Számokat már tudunk ábrázolni! 2-es számrendszer előnyben!
- Mi a helyzet a karakterekkel? Fontos ez?
- Kódolás: kód(Code),francia eredetű szó, rejtjelhez köthető, információt hordozó szimbólum, olyan módszer ami szimbólumokat és azok értelmezését összekapcsolja!
  - Kódolás, dekódolás (rejtjelezés, visszafejtés) – régi eszköztár
- Számítógép világában természetes módon számok léteznek -> Létre kell hozni egy szám-karakter hozzárendelési táblázatot!
  - Ezzel megszülettek a karakter kód táblázatok: ASCII, UTF-8, stb.

# Információ tárolás eszközei I.

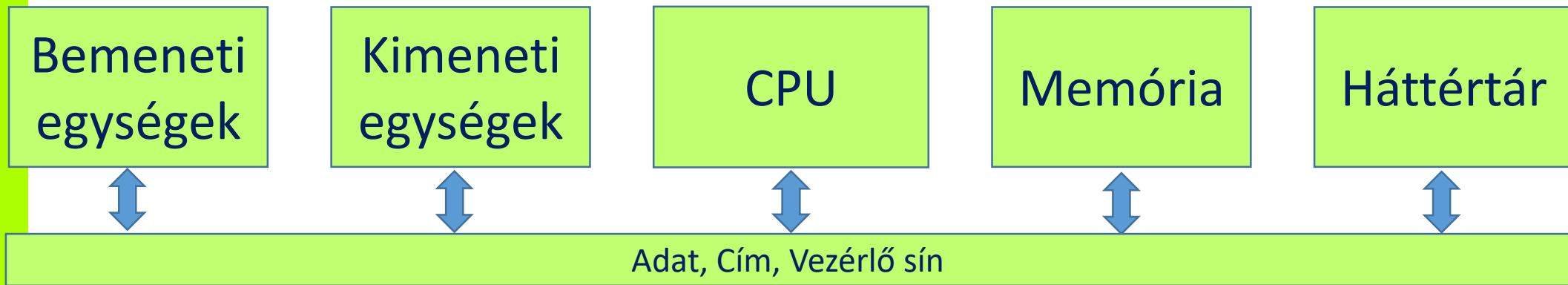
- Memória – a processzor utasításait, program adatait tárolja
- Memória típusok
  - RAM, SRAM, DRAM, DDR, DDR2,DDR3,DDR4,DDR5...
  - ROM, PROM, EPROM, EEPROM, FLASH
  - FLASH – ma kizárolag ez használatos, olcsó(bb), szilárdtest alapú
    - Működési elv: A flash tranzisztorban, cellában(1 bit) lévő elektron töltés hordozza az információt.
    - Két fő típusa van: NOR (Not OR), NAND (Not AND)
      - NOR – bárhonnan lehet írni,olvasni bájtokat, kényelmesebb, Intel fejlesztés
      - NAND- sorban memória blokkot lehet olvasni, gyorsabb, olcsóbb, Samsung fejlesztés

# Információ tárolás eszközei II.

- Mágnes szalag – ma is használt, elsősorban adatmentésre
- Mágnes lemez – ma még a leggyakrabban használt háttértár
  - FDD – Floppy Disk Drive
  - HDD - Hard Disk Drive
- Működési elv: Egy cella mágneses polaritása jelenti az információt.
- Optikai lemezek – CD, DVD, Blu-Ray
  - Működési elv: Visszaverődő fény(lézer) időkülönbsége hordozza az információt.(Pit-Land)

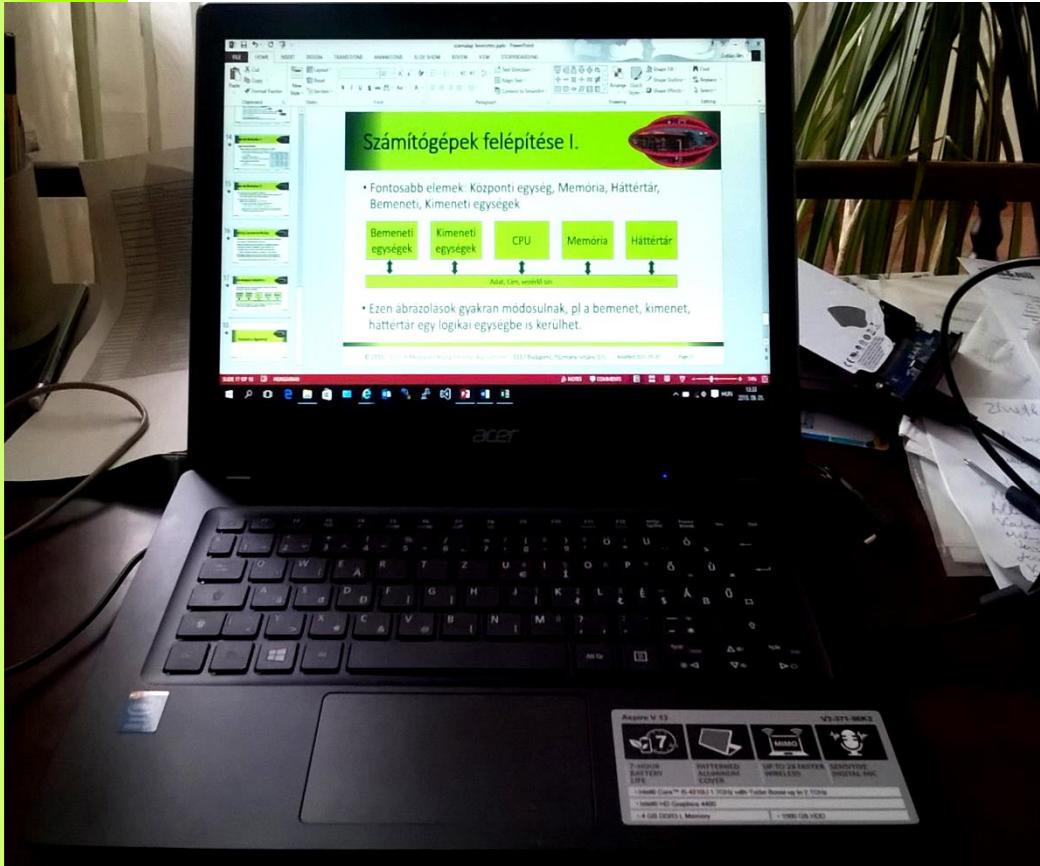
# Számítógépek felépítése I.

- Fontosabb elemek: Központi egység, Memória, Háttértár, Bemeneti, Kimeneti egységek



- Ezen ábrázolások gyakran módosulnak, pl a bemenet, kimenet, háttértár egy logikai egységbe is kerülhet.

# Számítógép kívül- belül



E1

# Hol találunk számítógépet?

- Loviban...😊
- Felhőben, telefonunkban, fényképezőben, televízióban, kamerában... mindenhol!
- Miben különböznek?
  - Feladatokban leginkább!
  - Általános számítógépek
  - Cél számítógépek
- Hardver, szoftver különbségek!

# Hardver-Szoftver különbségek

- Kliens-szerver gépek.
  - Kliens, jellemzően egy felhasználó igényeit kielégítő számítógép.
  - Szerver, jellemzően sok felhasználó kiszolgálását végzi!
- Hardver különbségek
  - Szerver esetén a klasszikus input/output eszközök hiányoznak!
  - Kliens esetén ez lényeges!
- Szoftver különbségek
  - Operációs rendszer
  - Egyéb felhasználói programok

# Operációs rendszerek

- Linux (SUSE, Ubuntu, Red Hat, Debian, stb)
  - UNIX-LINUX
- Apple OSX,iOS
- Windows (7,8,10), Win2012
- Felhasználói felületek
  - Grafikus
  - Karakteres
- A félév során a LINUX(UNIX) alaplehetőségeket nézünk meg!
  - Majd script programozunk!

# Kiszolgálók elérése I.

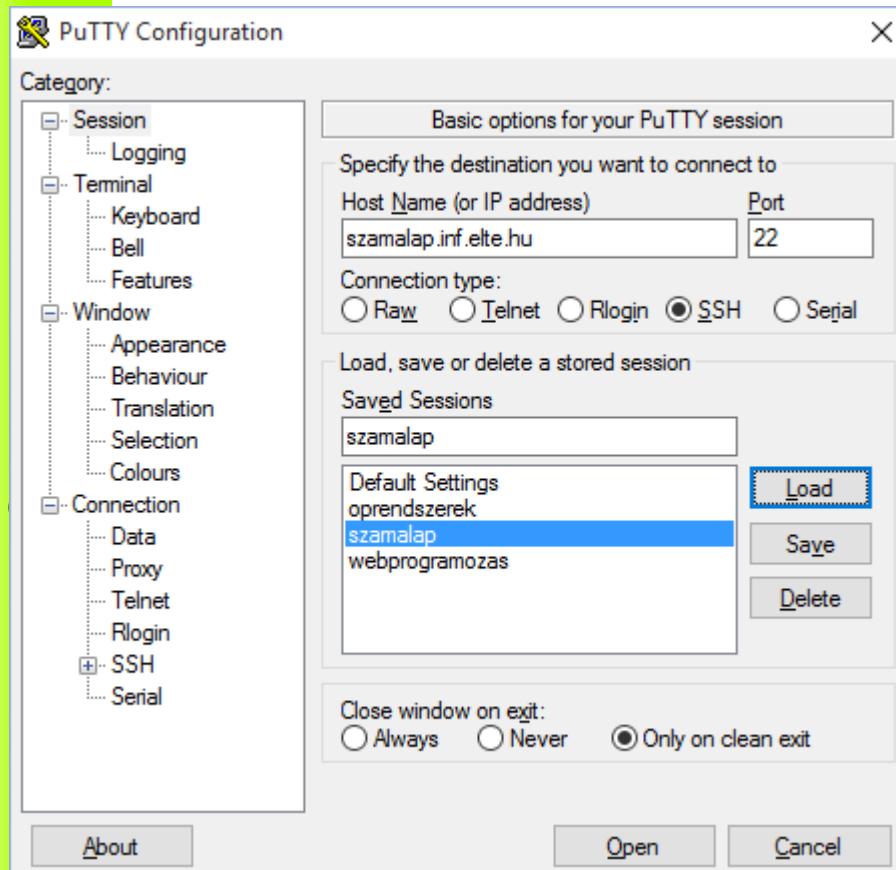
- Korábban csak a számítógépes termék termináljairól volt lehetséges!
- Ma „hálózati” kapcsolaton keresztül!
  - A terminál szobák megszűntek...:(
- Hálózati kapcsolatot biztosító eszközök.
  - Soros, párhuzamos port, ma nem használt.
  - USB port, jellemzően speciális esetben használt.
  - Hálózati (ethernet) kártya(LAN), RJ-45 port, UTP(STP) kábel, 10/100/1000
  - Vezeték nélküli kártya(WIFI), IEEE 802.11 a/b/g/n/ac

# Kiszolgálók elérése II.

- Hálózati elérés biztonsága
  - Az alap szabványok jellemzően nem tartalmaznak titkosítást!
  - Például, HTTP, titkosított kapcsolatot a HTTPS használ.
- Karakteres elérés
  - Telnet – ma ritkán használt, mert nem titkosított kapcsolatot használ!
  - FTP – szintén titkosítatlan, fájl transzfer protokoll!
  - Titkosított kapcsolatot használ:
    - SSH vagy SSL alap
    - RSA (**Rivest-Shamir-Adleman**) aszimmetrikus kódolás.
- Grafikus kapcsolatok

# Terminálkapcsolat

- Putty.exe – [www.putty.hu](http://www.putty.hu) oldalról letölthető!

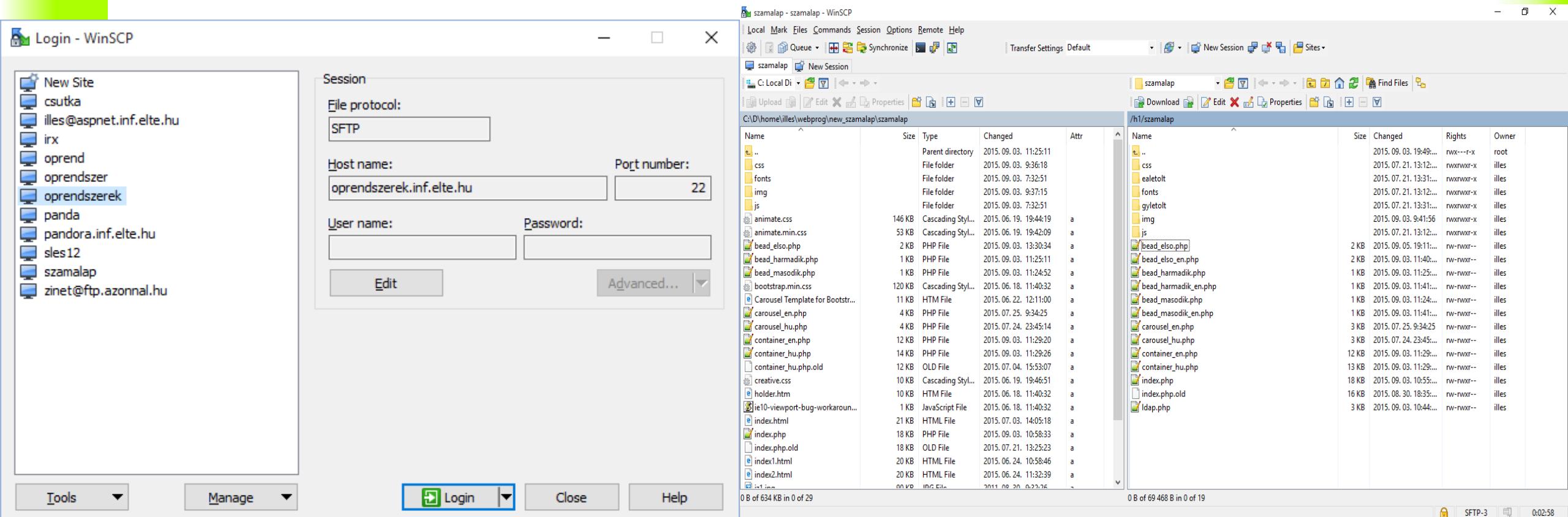


The screenshot shows a PuTTY terminal window titled 'szamalap.inf.elte.hu - PuTTY'. The terminal displays a Linux shell session. The user 'illes' is logged in and enters several commands:

```
illes@szamalap.inf.elte.hu's password:  
Linux pandora 4.0.9-grsec-pandora #1 SMP Fri Jul 31 18:02:34 CEST 2015 i686  
* Oracle adatbázis elérése lehetséges sqlplus cliensel a pandora-n.  
Használat: sqlplus username@oradb v sqlplus username@ablinux  
  
* Meglevo szövegfajlokat az iconv parancssal lehet konvertálni:  
iconv -f ISO-8859-2 -t UTF-8 <regi_iso.txt >uj_utf8.txt  
iconv -f UTF-8 -t ISO-8859-2 <regi_utf8.txt >uj_iso.txt  
  
UTF-8 ekezeteszt: áéíóööúü ÁÉÍÓÖÖÚÜ  
.....2013.05.24. PUTTY->afs home konyvtar elerese laborokbol!  
kinit usernev  
aklog ; cd  
....  
  
Disk quotas for user illes (uid 11264):  
Filesystem blocks quota limit grace files quota limit grace  
labhome.inf.elte.hu:/cluster/home  
32 256000 256000 6 0 0 0  
Volume Name Quota  
user.illes 10485760 1496 0% 0%  
Hajra Fradi!  
illes@pandora:~$
```

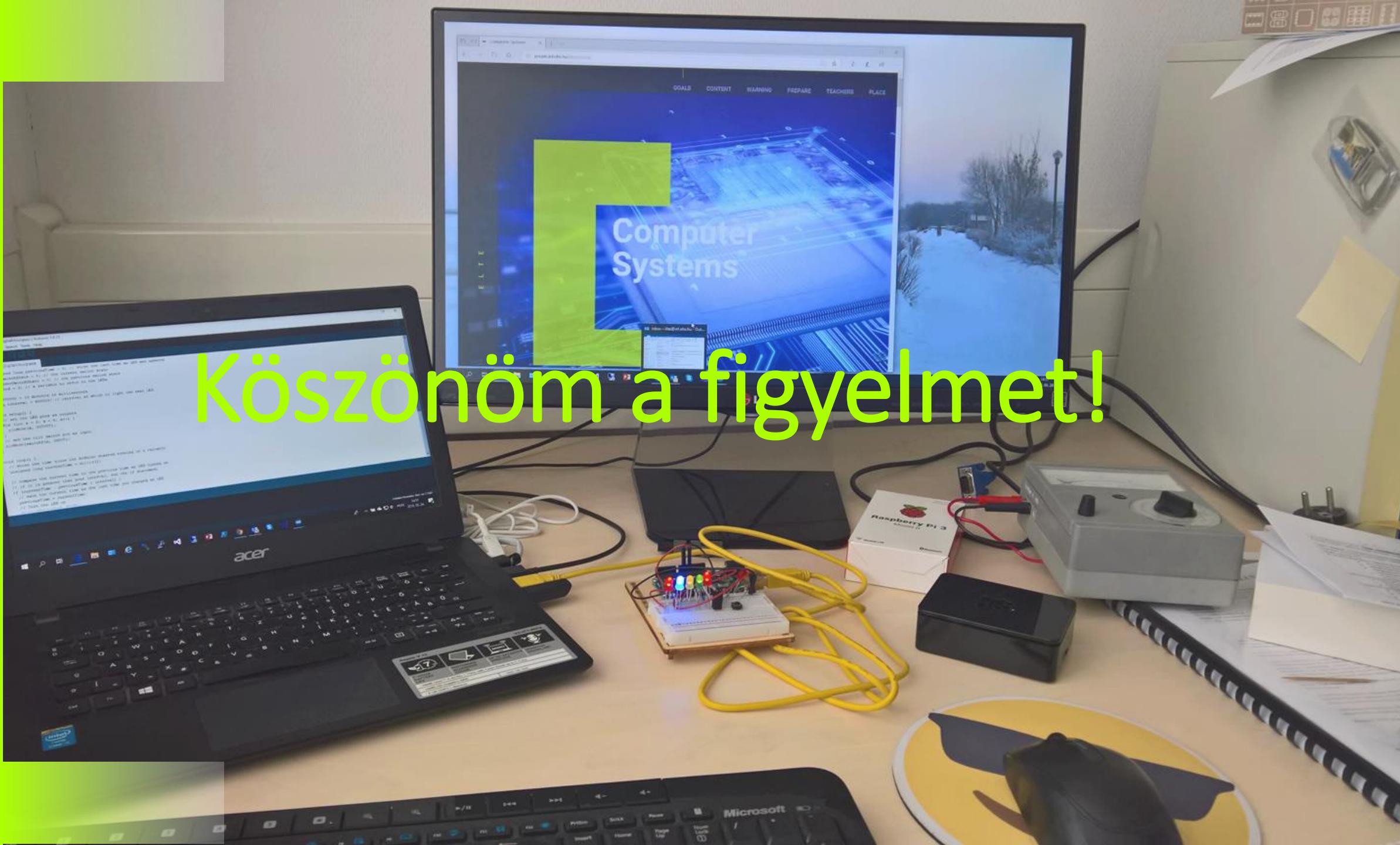
# Fájlok másolása

- winscp.exe – <http://www.winscp.net> oldalról letölthető!



# Összegzés

- Miről beszéltünk ma?
  - Számítógépek napjainkban
  - Alapvető felépítések
  - Jelek, tárolás, számábrázolás
  - Szoftver-hardver-operációs rendszerek



Köszönöm a figyelmet!