Pázmány Péter
Katolikus Egyetem

Bevezetés a számítástechnikába

PPKE, ITK, 2016. őszi félév

Papp Áron

1. óra
Bemutatkozás, bevezetés

Bemutatkozás

- A tantárgy felelőse: Dr. Siklósi Borbála
  Email: siklosi.borbala@itk.ppke.hu
  http://users.itk.ppke.hu/~sikbo
  Szoba: 314
- Gyakorlatvezetők:
  - Nika Zsolt
    Email: nika.zsolt@itk.ppke.hu
    http://users.itk.ppke.hu/~nikzs
  - Papp Áron
    Email: papp.aron@itk.ppke.hu
    http://users.itk.ppke.hu/~papar

Laborrend

Csoport	Idő / hely	Gyakorlatvezető
06	2016/17/1 K:14:15-16:00(ITK 422 PC labor)	Nika Zsolt
05	2016/17/1 K:12:15-14:00(ITK 422 PC labor)	Papp Áron
07	2016/17/1 K:10:15-12:00(ITK 422 PC labor)	Papp Áron
01-02MB	2016/17/1 K:12:15-14:00(ITK 219 PC labor)	Dr. Siklósi Borbála
03-04MB	2016/17/1 K:10:15-12:00(ITK 219 PC labor)	Dr. Siklósi Borbála
DB	2016/17/1 H:10:15-12:00(ITK 219 PC labor)	Dr. Siklósi Borbála
08	2016/17/1 K:10:15-12:00(ITK 302 PC labor)	Nika Zsolt

Számonkérés

- ZH-k
  - 2 db lesz, mindkettő max. 100 pont
  - Az első az őszi szünet környékére fog esni
- RöpZH-k
  - Minden óra elején, “az előző rész tartalmából”
  - Pontozás: -1 / 0 / +1
- Aláírás feltétele
  - Max. 3 hiányzás
  - RöpZH-k pontszámainak összege pozitív legyen
  - Mindkét ZH-t min. 50%-ra kell megírni
    (az egyiket lehet pótolni)

SZÁMONKÉRÉS

- Értékelés
  - 51 - 60 : 2
  - 61 - 70 : 3
  - 71 - 85 : 4
  - 86 - : 5
- Programozási feladat
  - Választható
  - A feltételek teljesülése esetén a végső pontszámhoz max. 10 ponttal járul hozzá
  - bash nyelven kell elkészíteni
- Órai aktivitás
  - Plusz ponttal jutalmazzuk :)

A félév rendje

1. Bemutatkozás, bevezetés, ITK-s alapok
2. Számrendszerek, számábrázolás, karakterkódolás
3. Számítógép generációk, operációs rendszerek
4. Hardver alapismeretek
5. Adattárolás és fájlrendszerek
6. Linux alapok
7. Linux alapok - folyt.
8. LaTeX 1.
9. LaTeX 2.
10. LaTeX 3. MB
11. LaTeX 3. DB
12. Szöveget tartalmazó fileok
13. Hálózatok

Géptermi házirend

- A gépteremben csak a kurzusra feliratkozott hallgatók tartózkodhatnak
- Enni, inni tilos!
- Gyakorlatilag minden tilos, csak a gépek rendeltetés szerű használata engedélyezett
- Szándékos rongálás (hardware / software) esetén fegyelmi eljárás kezdeményezhető
- Bejelentkezéshez mindenki az egyetemi azonosítóját használja

Secure Shell

- Mi az SSH?
  - A Secure Shell egy szabványcsalád, és egyben egy protokoll is, amit egy helyi és egy távoli számítógép közötti biztonságos csatorna kiépítésére fejlesztettek ki
  - Nyilvános kulcsú titkosítást használ a távoli számítógép hitelesítésére
  - Opcionálisan a távoli számítógép is hitelesítheti a felhasználót
- Putty
  - Terminál emulátor
  - Különböző protokolokat támogat, mi SSH-n fogjuk elérni a szervert
- WinSCP
  - Secure copy
  - Fájlátvitel SSH-n keresztül

GNU/Linux

- A GNU egy komplett operációs rendszer
- A Linux egy UNIX-szerű, nyílt forráskódú rendszermag
- A GNU projectet Richard Matthew Stallman 1983-ban indította
- A Linux-ot Linus Torwalds kezdte fejleszteni 1991-ben
- Különböző elektronikus eszközökben megtalálható
  - Hálózati eszközök (routerek, switchek, …)
  - Hordozható eszközök (mobiltelefonok, okostelefonok, PDA-k, tabletek, órák, navigációs készülékék, …)
  - Háztartási gépek, szórakoztató elektronikai berendezések, konzolok, set-top boxok, …)
  - Szerverek, személyi számítógépek
  - Szuperszámítógépek

2. Óra
Számrendszerek, számábrázolás, karakterkódolás

A számrendszer

- A számrendszer [numeral system] matematikai fogalom: egy módszer, melynek célja az írott formában történő megjelenítés
- Helyiértéken (pozíción) alapuló számrendszereket tárgyalunk (a római számok például sorrendiségen alapulnak)
- A számrendszer alapja [base, radix] meghatározza az egyes pozíciókba írható számjegyek maximumát
- A számjegyek [digit] a számok írására használt karakterek
- A számítástechnikában gyakran használt számrendszerek: 2, 8, 10, 16

Alaki- és helyiérték

- Számjegy értéke = alaki érték · helyiérték
- Alaki érték: számjegyhez tartozó érték
- Helyiérték: a számrendszer alapjának pozíció szerinti hatványa

Egész és tört számok

- Egész számok felírása
  $a_n\ a_{n-1}\ a_{n-2}\ \dots\ a_1\ a_0$
- Egész számok értéke
  $(a_n \cdot A^n) + (a_{n-1} \cdot A^{n-1}) + \dots + (a_1 \cdot A^1) + (a_0 \cdot A^0)$
- Tört számok felírása
  $a_n\ a_{n-1}\ \dots\ a_1\ a_0\ a_{-1}\ \dots\ a_{-k}$
- Tört számok értéke
  $a_n \cdot A^n + a_{n-1} \cdot A^{n-1} + \dots + a_1 \cdot A^1 + a_0 \cdot A^0 + a_{-1} \cdot A^{-1} + \dots + a_{-k} \cdot A^{-k}$

Átváltás 10 -> 2 számrendszerbe

```
177 | 1
 88 | 0
 44 | 0
 22 | 0
 11 | 1
  5 | 1
  2 | 0
  1 | 1
  0 |
```

$$177_{(10)} = 1011\ 0001_{(2)}$$

Pontosság

- Nem egész számok nem mindig írhatóak fel véges számjeggyel
- Nem egész számok pontossága függ a számrendszer alapjától
- Pl.:

  $1/3_{(10)} = 0.333\ldots_{(10)} = 0.1_{(3)}$

- Keressünk további példákat!

Az adatmennyiség mértékegységei

- Alapegység: bit (true/false)
- 1 Byte = 8 bit

SI		Bináris	
Prefix	**Szorzó**	**Prefix**	**Szorzó**
K (kilo)	1000	Ki (kibi)	1024
M (mega)	1000^2	Mi (mebi)	1024^2
G (giga)	1000^3	Gi (gibi)	1024^3
T (tera)	1000^4	Ti (tebi)	1024^4
P (peta)	1000^5	Pi (pebi)	1024^5

Gépi számábrázolás

- Nem negatív egész számok: megegyezik a bináris számok leírásával, tehát 2-es számrendszerben tároljuk
- Összeadás művelete nemnegatív egész bináris számokon

```
  1001 1011
+ 0110 1100
-----------
1 0000 0111
```

- Hogyan dönthetjük el, hogy melyik bináris szám nagyobb?

Gépi számábrázolás

- Negatív számok előjelbites ábrázolása
  - A legnagyobb helyiértéken a szám előjelét tároljuk
  - A maradék helyiértékeken a számot
  - Pl. 8 bites változóméret esetén: $-32_{(10)} = 1010\ 0000_{(2)}$
- 2-es komplemens ábrázolás
  - A szám abszolút értékének a számjegyeit invertáljuk
  - Majd hozzáadunk 1-et a számhoz
- A 2-es komplemens előnye: nincs szükség kivonás műveletre
  - Pl. : $19_{(10)} - 9_{(10)}$ $0001\ 0011_{(2)} - 0000\ 1001_{(2)}$

```
  Kivonandó: 0000 1001      0001 0011   A túlcsordulást levágva az
1-es kompl.: 1111 0110     +1111 0111   eredmény: 1010 (2) vagyis 10 (10)
2-es kompl.: 1111 0111     ----------
                          1 0000 1010
```

MSB/LSB

- MSB: Most Significant Bit
- LSB: Least Signinficant Bit
- Architektúránként eltér
- Azt jelzi, hogy a fizikai tárolás vagy hálózati továbbítás során a változóhoz tartozó adott méretű memóriaterület címéhez tartózó Byte a számnak a legértékesebb vagy a legkevésbé értékes helyiértékét jelöli
- Endianness: big-endian / little-endian
  (-> Swift: Guliver utazásai)

Túlcsordulás

- Egész számok ábrázolása esetén meghatározott számú biten korlátozott az ábrázolható számok nagysága
- Lehetséges alul- vagy túlcsordulás
- A megvalósítástól függően eredményezhet
  - levágást: a nem ábrázolható rész levágjuk, nem használjuk fel
  - szaturációt: a legnagyobb vagy legkisebb ábrázolható értéket tároljuk

Lebegöpontos számábrázolás

- Normalizált alak: a szám felbontása egy kéttagú szorzatra, ahol a második tag a számrendszer alapjának valamely hatványa, melyet, ha az első számmal beszorzunk az eredeti számot kapjuk vissza. A tizedes pont előtt csak 1 számjegy lehet.
  Pl. : $380_{(10)} = 3.8 * 10^2$
- Az így kapott eredményt bináris formában tároljuk el, és a legértékesebb helyiértéken jelőljük a szám előjelét
- IEEE 754: {előjel} {exponens} {mantissza}
- Bináris szám esetén a tizedespont előtt csak 1-es lehet
- NaN: Not a Number

Karakterek és karakterkészletek

- A karakter a számhoz hasonlóan fogalom, a karakterkészlet egy kódtáblázat, ahol az egyes karakterekhez kódszámot rendelnek
- ASCII kódtábla
  - American Standard Code for Information Interchange (1960-as évek)
  - Alapesetben 7-bites, az extended változat 8-bites
- Unicode
  - A cél egy kódtáblában tárolni a világ összes betűjelét
  - Az Unicode egy kódolási szabvány, nem kódtábla
- UTF-8
  - Változó 1-6 Byte hosszú tárolás
  - ASCII kompatibilis és önszinkronizáló (nem kell a string elejéről kezdeni az olvasást, hogy elkülönüljenek a karakterek)

3. Óra
Operációs rendszerek

Operációs rendszer fogalma

- Ajánlott irodalom: Tanenbaum - Operációs rendszerek
- A számítógépet közvetlenül gépi nyelv szintjén programozhatjuk, ez azonban kényelmetlen
- Minden alrendszert (utasításkészlet, memóriaszervezés, I/O rendszer, sínstruktúra) a programozónak kellene lekezelnie a programjában, de nem akarja
- Az operációs rendszer elrejti előlünk a hardvert: absztrakciós rétegeket hoz létre, hogy a hardver elérése a felhasználói programokból egyszerű legyen
- Kiterjesztett vagy virtuális gépet biztosít a felhasználónak

Operációs rendszer mint eröforrás-kezelö

- “felülről lefelé” nézőpont: kényelmes csatlakoztatási felület a felhasználók számára
- “alulról felfelé” nézőpont: az operációs rendszer célja, hogy az összetett rendszer minden részét kezelje
- Erőforrások megosztása:
  - időalapú: az erőforrások felváltott használata
  - téralapú: az erőforrás részekre osztása
- A hacker támadások egy része azt használja ki, hogy az adott program átlépi a saját hatáskörét, és hozzáfér egyéb folyamatokhoz rendelt erőforrásokhoz

Operációs rendszerek történelme I.

- Számítógép generációkon keresztül tekintjük at az op.rsz-ek fejlődését
- I. generáció (1945-1955): Vákumcsövek és kapcsolótáblák
- Howard Aiken, Neumann János, J. Presper Eckert, John William Mauchley és Konrad Suse számítógépek építésében értek el sikereket
- Először relék, majd vákumcsövek alkották a gépeket
- Programozás gépi nyelven történt, nem voltak programozási nyelvek sem
- Eleinte kapcsolótáblákat cserélgették, majd az 50-es években megjelentek a lyukkártyák

Operációs rendszerek történelme II.

- II. generáció (1955-1965): tranzisztorok és kötegelt rendszerek
- Ma ezeket a gépeket nevezzük mainframe-eknek
- Assembly, FORTRAN nyelvek használata
- Lyukkártyán, később szalagon vitték be a programokat
- Az eredmény a nyomtatóra ment
- Tudományos és mérnöki számításokra használták, pl. partiális differenciálegyenletek numerikus megoldására
- Op.rsz-ek: Fortran Monitor System és IBSYS (IBM rendszere a 7094-re)

Operációs rendszerek történelme III.

- III. generáció (1965-1980): integrált áramkörök
- A korábbi szó-orientált gépek mellett megjelentek a karakter-orientált gépek (pl. 1401), melyeket bankmok és biztosítók használtak szalagrendezésre, nyomtatásra
- A két irányvonalat az IBM egy közös rend-szerrel a System/360-nal egyesítette
- Később megjelent a 370, 4300, 3080, 3090, és a mai is kapható System Z
- Multiprogramozás: amig a processzor egy művelet eredményét várt, azalatt a memória egy másik részében számítást végzett

Operációs rendszerek történelme III.

- Spooling (Simultaneous Peripheral Operation On Line): a kártyákról a feladatokat lemezre másolták, így amikor egy feladat befejeződött a háttértárról gyorsan be tudta olvasni a következőt
- Időosztás (Time sharing): a processzor órajelén egyszerre több felhasználó/alkalmazás osztozik
- LAN (Local area network): helyi hálózat tette lehetővé a fájlkiszolgáló szerverek működését
- Közvetítő réteg (Middleware): a lokális felhasználókat kötötte össze a távoli erőforrásokkal

Operációs rendszerek történelme III.

- Mindeközben megjelentek a miniszámítógépek is, a DEC PDP sorozat jegyében, melyeket a korábbi nagygépek árának töredékéért lehetett megvásárolni
- Ken Thompson a Bell Labs munkatársa egy PDP-7-en kezdett el programozni egy egyfelhasználós rendszert, ami a Unix operációs rendszer t torkollott
- A Unix forrása nyílt volt, mindenki saját, inkompatibilis változatot kezdett fejleszteni
- Két fő változat: System V és a BSD
- Az IEEE POSIX nevű szabványát a legtöbb mai Unix betartja
- “Unix: Live free or die”

Operációs rendszerek történelme IV.

- IV. generáció (1980-): személyi számítógépek
- Az LSI (Large Scale Integration) áramkörök fejlődésével megérkezett a mikroprocesszor alapú személyi számítógépek kora: bárkinek lehetett saját gépe
- 8 bitesek: Intel 8080 (1974-ben jelent meg), Zilog Z80 (CP/M rendszert futtatott), Motorola 6800, MOS 6502
- 16 bitesek: Intel 8086, Intel 8088 (IBM PC)
- A Microsoft felvásárolt a DOS rendszert és MS-DOS néven adta ki, ami gyorsan elterjedt az IBM PC-ken
- Az MS-DOS tartalmazott a BASIC nyelv támogatást

Operációs rendszerek történelme IV.

- Doug Engelbart vetette fel pár évvel a DOS előtt a grafikus felhasználói felület ötletét (GUI), és Steve Jobs látta meg ebben a lehetőséget: 1984-ben jelent meg az Apple Macintosh
- Az első Mac a Motorola 68000 processzorát tartalmazta
- Később átváltottak az IBM 32, majd 64 bites RISC processzraira (PowerPC) és megjelent a Berkely Unix-ra épülő Mac OS X
- 2005-ben bejelentette az Apple, hogy átáll az Intel CPU-kra
- A Microsoft piacra dobta a Windows-t, hogy versenyben maradjon az Apple-el szemben

Operációs rendszerek történelme IV.

- Amikor a Unix forrását lezárta az AT&T a hallgatók elől Prof. Tanenbaum hozzáfogott a Minix fejlesztéséhez
- A Minixet kisméretűnek tarották meg, hogy a hallgatók is tudják futtatni gépeiken, így tett az egyik hallgató Linus Torvalds is
- Torvaldsnak hiányzott pár funkció a Minixből, amikre programot írt, majd egy másfajta terminalmeghajtót is készített, később egy lemezmeghajtót és fájlrendszert
- Az eredményeket a USENET-en comp.os.minix csoportban közzétette, és segítőkre lelt
- 1994. március 13-án megjelent a Linux 1.0 :)

Fogalmak: processzus

- Végrehajtás alatt lévő program
- Hozzá tartozik egy címtartomány (a memória egy szelete), amin belül a processzus írhat/olvashat
- Van egy regiszterkészlete is (utasításszámláló, veremmutató, ...)
- Egy adott processzushoz tartozó információkat az op.rsz. a processzustáblázatban tárol, és ezeket az adatokat használja fel, amikor az időosztás során az adott processzusnak újból ad egy CPU időszeletet
- A processzusok fa-strukturát alkotnak (szülő és gyerek processzusok)

Fogalmak: fájlok

- Az op.rsz. feladadata, hogy az I/O műveletek felett egy fájlrenszer absztrakciót biztosítson
- Rendszerhíváokkal lehet fájlokat létrehozni, törölni, olvasni és írni
- A fájlok könyvtárakba vannak szervezve és POSIX rendszereken valamennyi fájl az útvonala segítségével elérhető az ún. gyökérkönyvtárból
- Jogkezelés: owner, group, other / read, write, execute
- Specifikus fájlok: block- és karakterspecifikus lehet
- Adatcső: az egyik processzus kimenetét a másik bemenetére irányítja

Fogalmak: parancsértelmezö

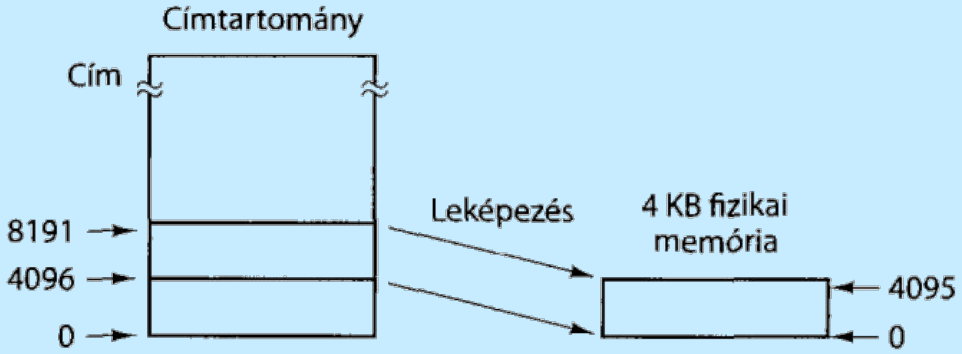
- Kapcsolódási felület a felhasználó és az operációs rendszer magja között
- POSIX rendszereken shell-nek hívják
- Többféle változatok: sh, csh, zsh, ksh, bash, ...
- A prompt jelzi, hogy az értelmező várja az utasítást
- A grafikus felhasználói felületek is gyakorlatilag parancsértelmezők

FOGALMAK: RENDSZERHÍVÁSOK

- A rendszerhívásokkal a felhasználói programok jelzik a rendszermag számára, hogy feladatot kell végrehajtson
- Gyakorlatilag olyan eljáráshívások, amik a magba más privilegizált op.rsz. komponensbe tudnak belépni
- Csoportjai:
  - Processzuskezelő rendszerhívások
  - Szignálkezelő rendszerhívások
  - Fájlkezelő rendszerhívások
  - Könyvtárkezelő rendszerhíváok
  - A védelem rendszerhívásai
  - Az időkezelés rendszerhívásai

FOGALMAK: VIRTUÁLIS MEMÓRIA

- Az 1950-es években a programozóknak muszáj volt akkora részekre bontani a programot, hogy elférjen a memóriában - ezt nevezzük átfedésnek (overlays)
- A módszer mai napig megmaradt, csak automatizálva lett - 1961-ben Fotheringham által (virtuális memória, lapozás)
- Ezáltal megtörtént a címtartomány és a memóriarekeszek fogalmának különválasztása



FOGALMAK: VIRTUÁLIS MEMÓRIA

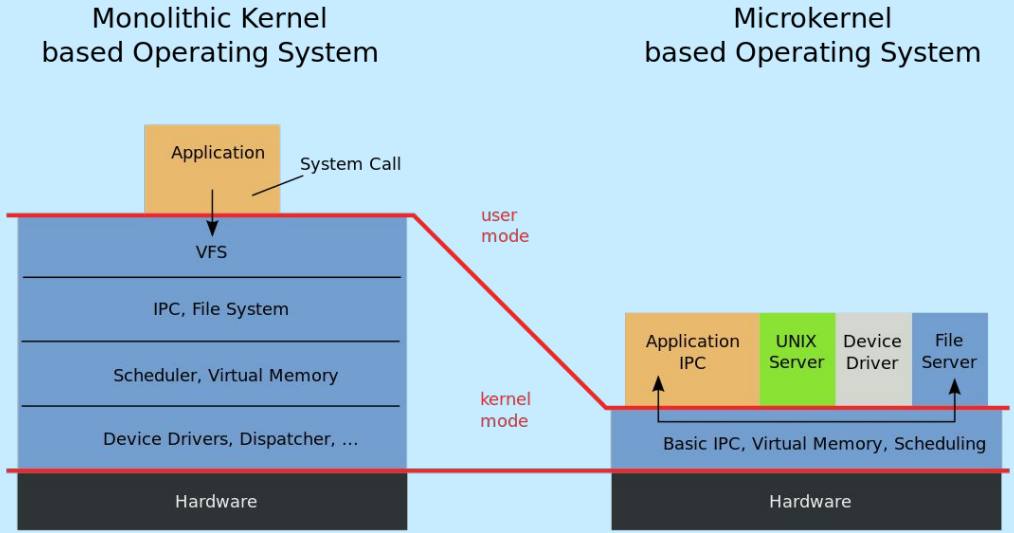
- A lapozás lépései:
  - A memória tartalmának lemezre mentése.
  - A 8192 és 12287 közti szavak megkeresése a lemezen.
  - A 8192 és 12287 közti szavak betöltése a memóriába.
  - A memóriatérkép megváltoztatása; a 8192 és a 12287 közti címek leképezése a 0 és 4095 közti memóriarekeszekre.
- Fogalmak:
  - A program a virtuális címtartományra hivatkozhat
  - A memórirekeszeket a fizikai címtartomány címzi meg
  - A memóriatérkép az egyes virtuális címeknek megfelelő fizikai címeket határozza meg

Struktúrák: monolítikus rendszerek

- “Struktúrája a struktúrálatlanság”
- Az operációs rendszer eljárások gyűjteménye, bármelyik hívhatja a másikat korlátozás nélkül
- A paraméterek és a visszaadott érték alapján minden eljárásnak jól definiált felülete van, ha a programozó úgy gondolja, hogy eljárásában egy másik eljárás valami hasznosat nyújthat, akkor azt szabadon hívhatja

Struktúrák: microkernel

- Szemben a monolítikus rendszerekkel a mag méretét minimalizálják, és külső forrásból éri el a kernel azokat a komponenseket, amiket nem tartalmaz



Struktúrák: rétegelt rendszerek

- A rendszert rétegekből álló hierarchia jellemzi, minden réteget az alatta lévőre építenek fel
- Első: THE (E. W. Dijkstra)
- Rétegek:
  - 5: Gépkezelő
  - 4: Felhasználói programok
  - 3: Bement/kimenet kezelése
  - 2: Gépkezelő processzus kommunikáció
  - 1: Memória- és dobkezelés
  - 0: Processzor-hozzárendelés és multiprogramozás

Struktúrák: virtuális gépek

- Egy adott harvder/szoftver architektúra emulálását jelenti
- Első: VM/370 (1979): különválasztották a multiprogramozást és a hardver eléréséhez használt kiterjesztett gépet
- Példák:
  - MS-DOS rendszerek futtatása Windowson emulált környezetben
  - VMWare, VirtualBox
  - Xen, LXD, WMWare, KVM
  - Java Virtual Machine, Python runtime
  - Linux chroot
  - Solaris containers
  - Android runtime

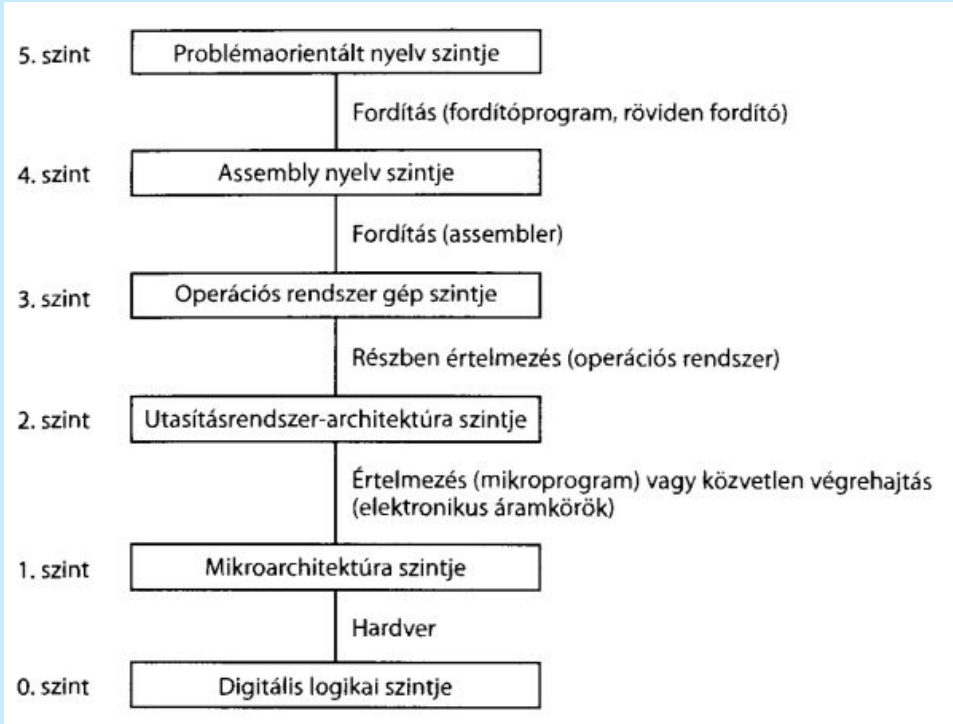
4. óra
Számítógép architektúra

Számítógép architektúrák

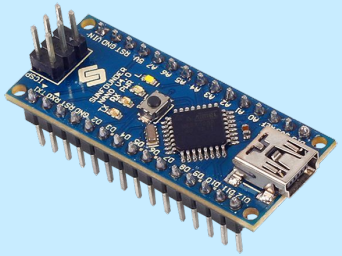
- Ajánlott irodalom: Tanenbaum - Számítógép architektúrák
- Digitális számítógép: problémák megoldása utasítások révén
- Program: utasítások sorozata
- Az elektronikus áramkörök utasítások egy szűk halmazát képesek felismerni, programjainkat konvertálni kell
- Fő utasítások:
  - Adj össze két számot
  - Ellenőrizd, hogy a szám nulla-e?
  - Egy számot másolj a memória egyik címéről egy másikra
- A gépi nyelvek kényelmetlenek az ember számára, ezért strukturálták a működést absztrakciók sorozatára, innen ered a struktúrált számítógép-felépítés

Többszintű számítógépek

- 5: Magas szintű nyelvek
- 4: Szimbólikus nyelv
- 3: Bővített utasítások
- 2: ISA
- 1: ALU
- 0: kapuk
- -1: elektronika

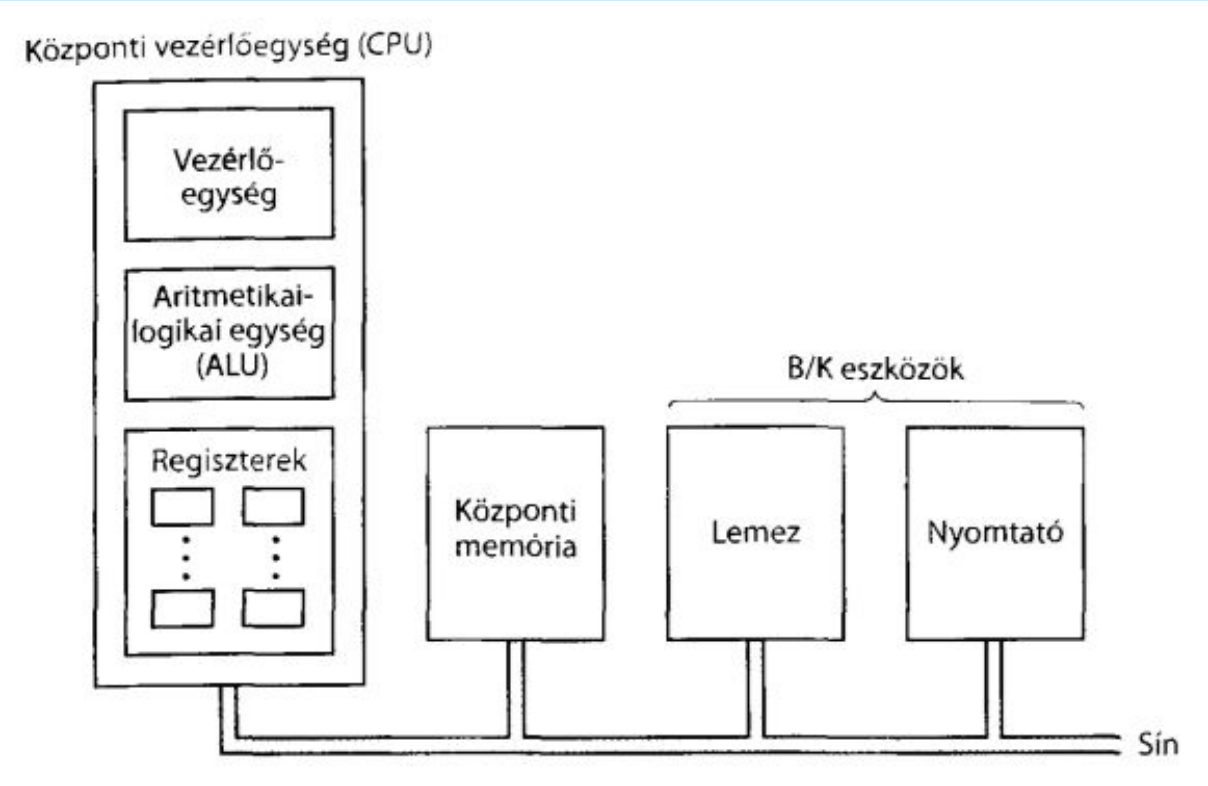


Számítógépek termékskálája



- Eldobható számítógép
- Mikrovezérlő
- Személyi számítógép
- Szerver
- Elosztott rendszer (klaszterek)
- Nagyszámítógép

Processzor: felépítés

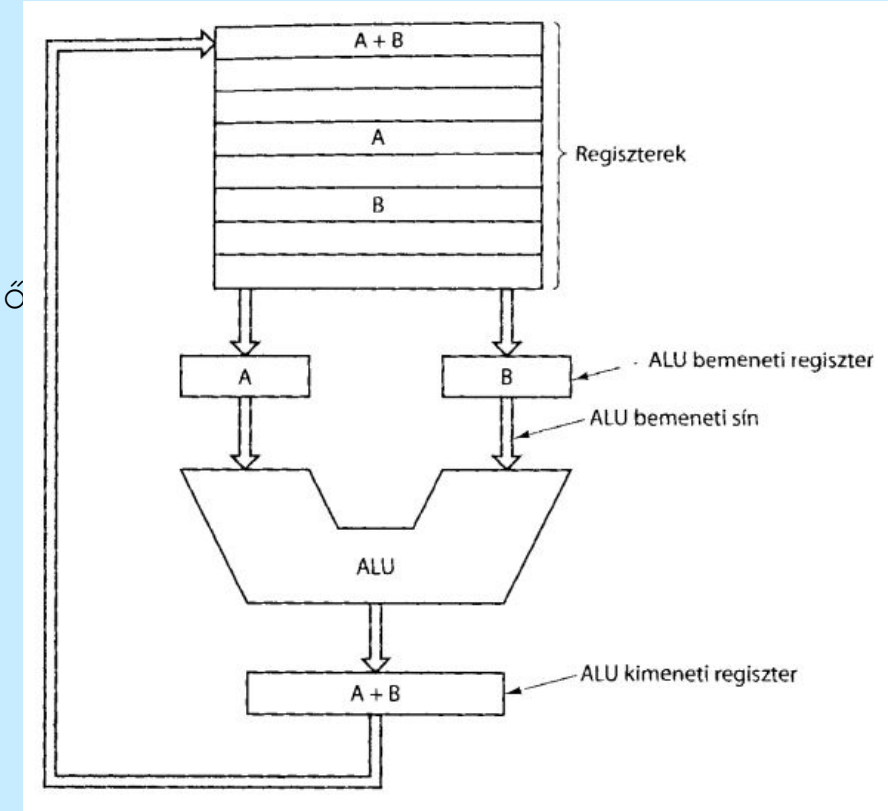


Processzor: felépítés

- Sín (Bus): összeköti a részegységeket, adatok és vezérlőjelek továbbítására szolgál
- CPU (Central Processing Unit): feladata a központi memóriában tárolt programok végrehajtása
- Regiszterek: kis méretű, gyors memória
- PC (Program Counter): az egyik regiszter, a következő program memória beli címét tartalmazza
- IR (Instruction Register): az utasításregiszter, a végrehajtás alatt lévő utasítást tartalmazza

Processzor: felépítés

- Adatút (Data path):
  - regiszterek
  - ALU (Aritmetical Logical Unit)
  - Sínek
- Az ALU bementi a regisz- terekből kimeneti regiszterekbe írja
- Fontosabb ALU műveletek
  - összeadás
  - kivonás
  - összehasonlítás nullával



Processzor: utasítás végrehajtás

- A végrahajtás lépései (betöltő-dekódoló-végrehajtó ciklus):
  - A soron következő utasítás beolvasása a memóriából az utasításregiszterbe
  - Az utasításszámláló beállítása a következő utasítás címére
  - A beolvasott utasítás típusának meghatározása.
  - Ha az utasítás memóriabeli szót használ, a szó helyének megállapítása.
  - Ha szükséges, a szó beolvasása a CPU egy regiszterébe.
  - Az utasítás végrehajtása.
  - Vissza az 1. pontra, a következő utasítás végrehajtásának megkezdése.

Processzor: CISC és RISC

- CISC: Complex instruction set computing
- RISC: Reduced instruction set computing
- #1: Program végrehajtására képes processzoráramkör építése
- #2: Programot értelmezni tudó interpreter írása
- Azt állították, hogy a számítógépek tervezésének legjobb módja, ha kevés egyszerű utasításunk van, amelyek adatútjának egyszeri bejárásával végrehajthatók.
- Ha egy CISC-utasítás helyettesítéséhez 4-5 RISC-utasítás kell, még mindig a RISC a gyorsabb, mert a RISC utasítások 10-szer gyorsabbak egy CISC-nél (mivel nem interpretáltak)

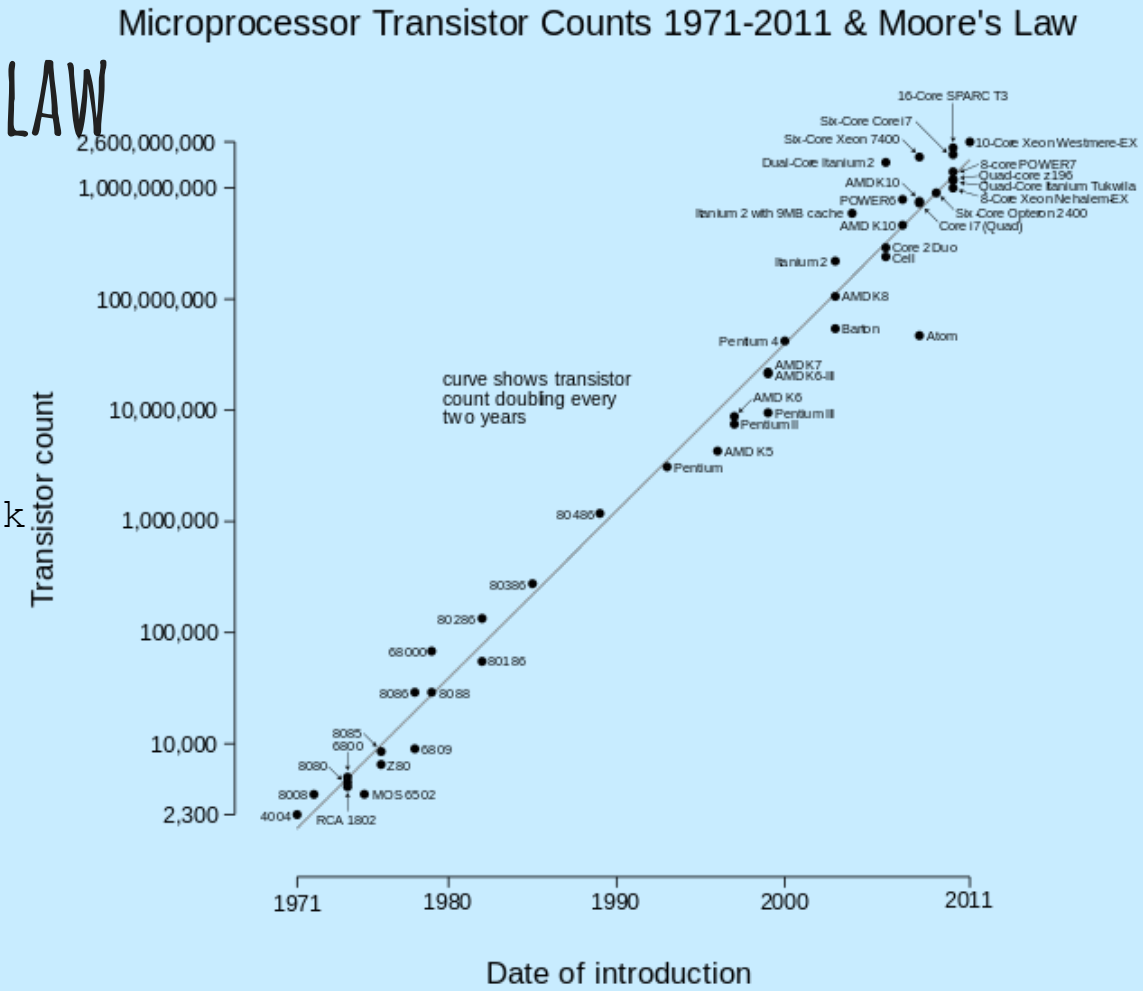
Processzor: CISC és RISC

- Ugyan logikusak az érvek a RISC mellett, mégsem szorította ki a piacról a CISC-et az alábbiak miatt:
  - Visszafelé kompatibilitás
  - Dollármilliárdok, amiket a CISC rendszerek fejlesztésére költöttek
  - 486-tól kezdődően az Intel egy RISC magot is épít a CPU-kba az egyszerű utasítások számára, a bonyolúltabbakat CISC módon hajtja végre
- A hibrid megközelítés nem olyan gyors, mint a tisztán RISC módszer, de versenyképes, és megmarad a kompatibilitás

Processzor: Moore's law

- Gordon E. Moore az Intel egyik alapítója

- "Az integrált áramkörök összetettsége 18 hónaponként megduplázódik"



Memória: bitek

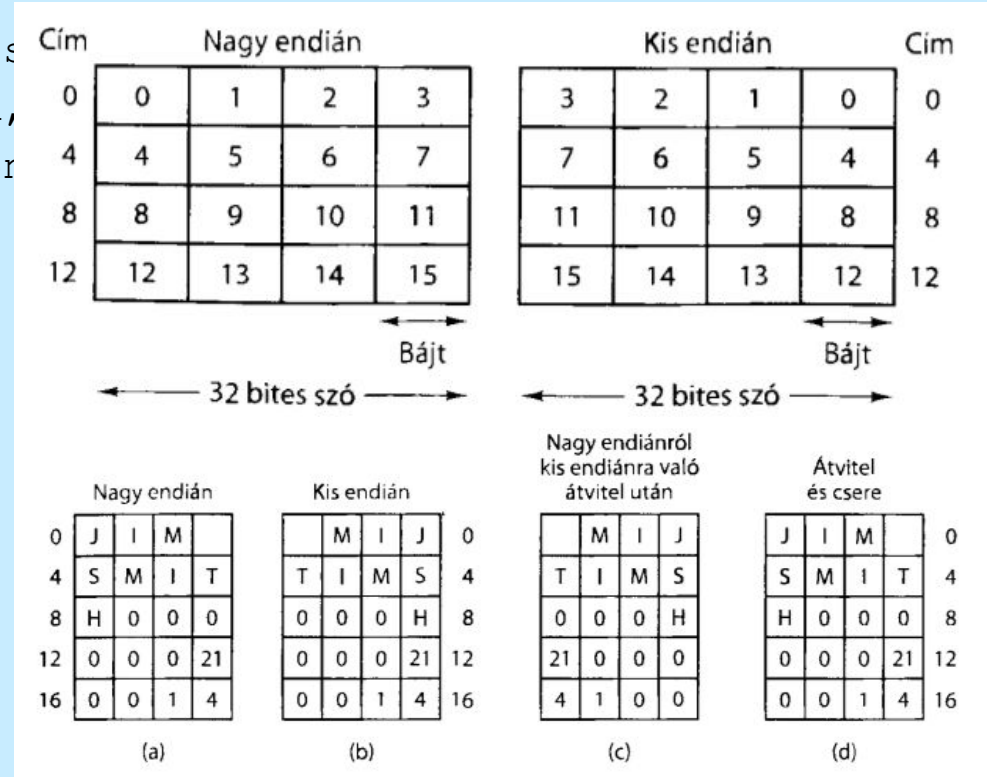
- A memória alapegysége a bit (0 vagy 1)
- Minél több feszültségszintet kell megkülönböztetnünk, annál bonyoltultabb feladat, ezért “hatékony” 2-es számrendszert használni
- BCD (Binary coded decimal): 4-biten tárol egy 10-es számrendszer beli számot
- 1944
  - decimális: 0001 1001 0100 0100
  - bináris: 0000 0111 1001 1000

Memória: címzés

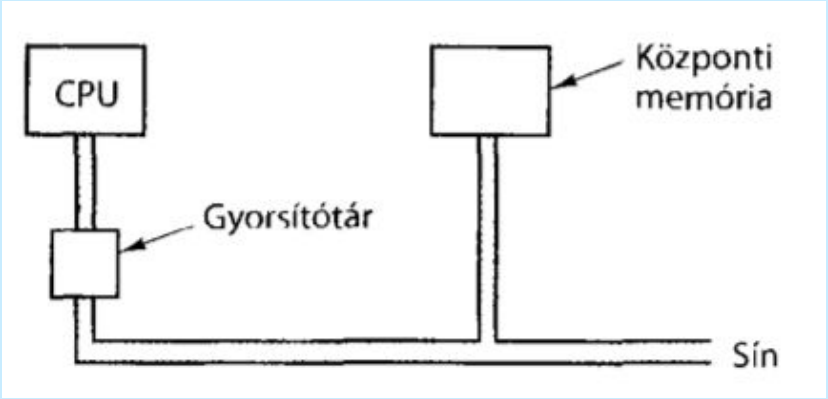
- A memória egyforma, k méretű cellákba van rendezve
- 0-tól n-ig címezhetőek a cellák - n*2^k bit
- A 8-bites cella-méret vált általánossá
- Egy 64 bites rendszernek 64 bitesek a regiszterei, így 64 biten tudja megcímezni a memóriát is (maximális memória méret: 2^{64} * 1 Byte)

Memória: bájtsorrend

- Mindkét reprezentáció teljes
- A problémák akkor kezdődnek,
  küldeni a másiknak hálózator



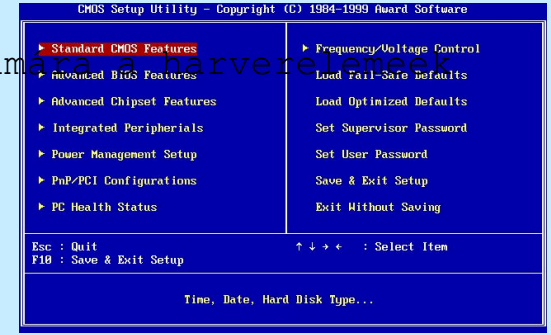
Memória: gyorsítótár



- CPU gyártók célja sebesség növelése, míg a memória gyártók a kapacitást növelik
- Gyors memóriát a CPU lapkán kell elhelyezni, ez azonban drága lenne, a sínen kapcsolt memória jóval olcsóbb
- Hibrid megoldás: kevés, gyors memória (cache) a CPU lapkán és sok, de lassú memória a sínen keresztül elérve
- Lokalitási elv: soron következő utasítások gyakran használják a korábbi memóriaterület szomszédságát
- A cache-be mindig egy területet másol, így esélyes, hogy egy következő utasítást csupán cacheből ki lehet szolgálni

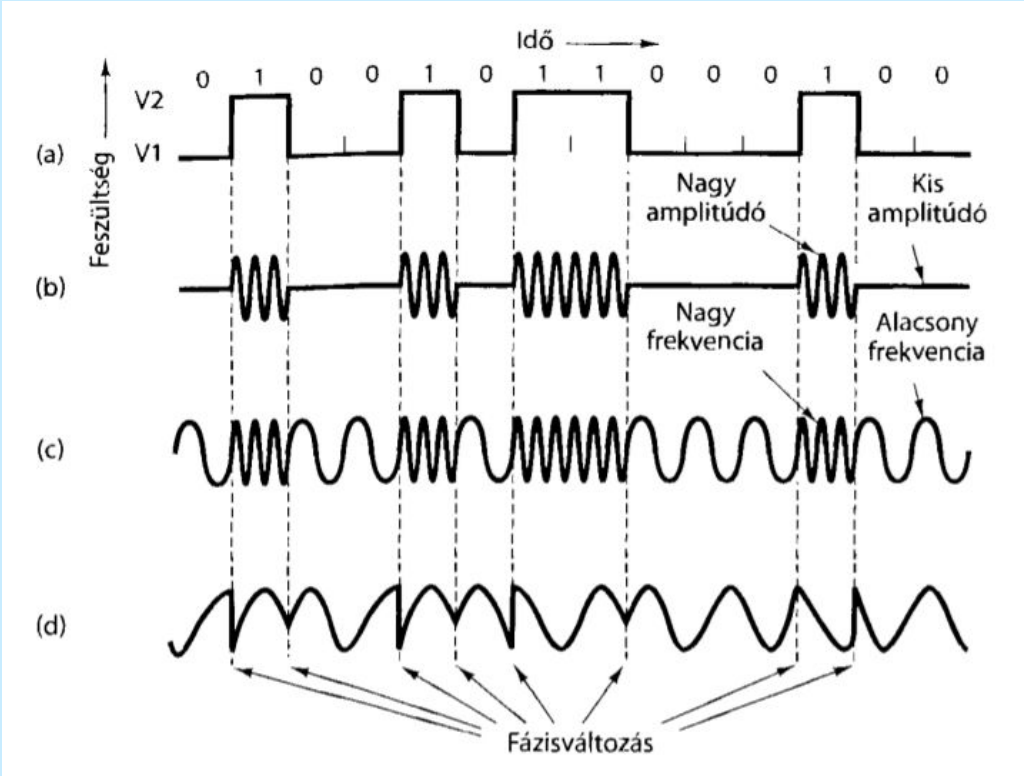
Basic Input Output System (BIOS)

- Az alaplapin túl a bővítőkártyák is saját BIOS-al rendelkezhetnek
- Korábban hasonló funkciót töltöttek be a firmware-ek, 1975-ben vezették be a BIOS-t, mint (könnyen) módosítható változatot
- Hardver és a szoftver közötti kapcsolat
- Feladatai:
  - Hardver ellenőrzése
  - Hardvervezérlők betöltése
  - Operációs rendszer betöltése
  - Interfész biztosítása az operációs rendszer számára a hardverelemek eléréséhez



Fizikai jelátviteli módszerek

a) Kétszintű jel
b) Amplitúdómoduláció
c) Frekvenciamoduláció
d) Fázismoduláció

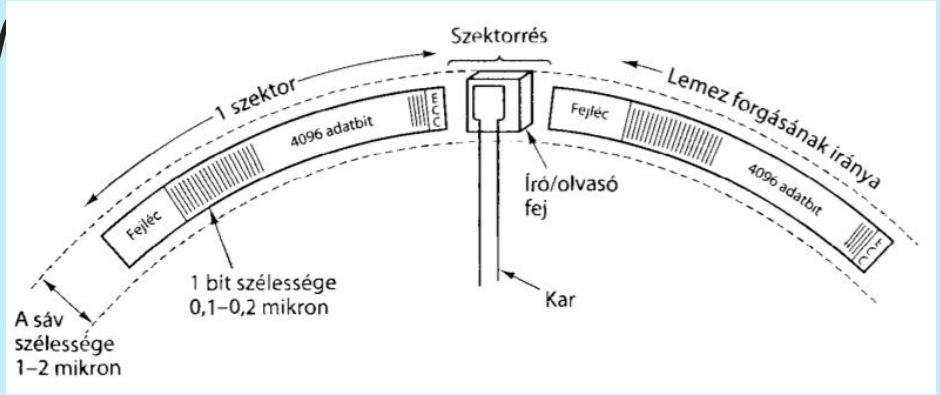


5. Óra
Adattárolás

Háttérmemória: hiearchia

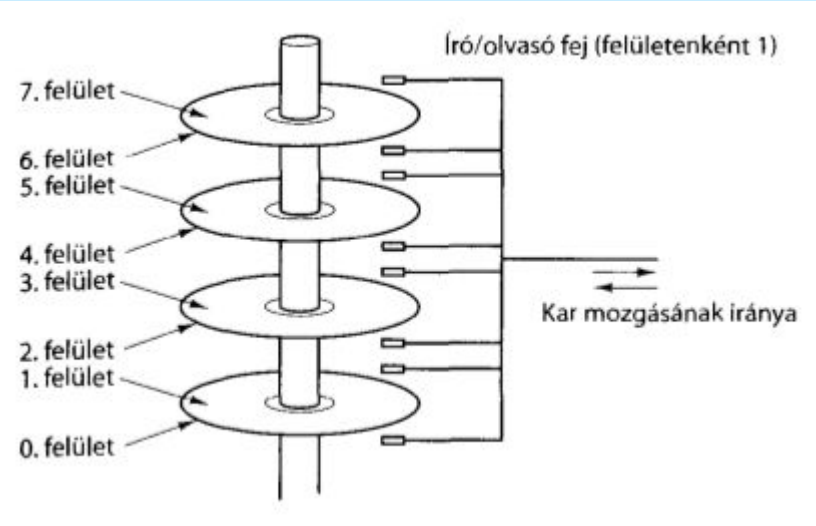
- Egy bájt tárolásának költsége
- A sebesség is fentről lefelé c
- Jellemző elérési idők
  - Regiszterek: 1-5 ns
  - Memória: 10-50 ns
  - SSD: 0.1-0.3 ms
  - Mágneslemez: 3-12 ms

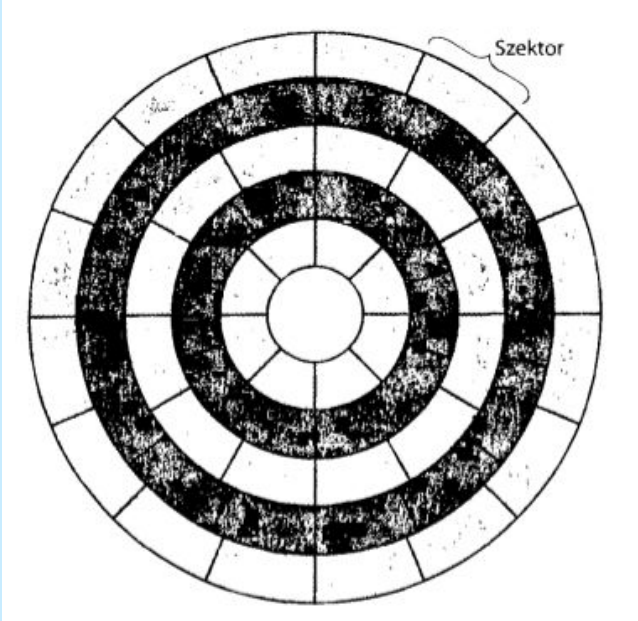
Háttérmemória: merevlem



- Alumínium lemez, mágnesezhető bevonat-tal ellátva
- Indukciós tekercset tartalmazó fej a lemez felszíne felett lég-párnán lebeg
- Egy teljes körülfordulás alatt felírt bitsorozat a sáv
- A sávok szektorokra vannak bontva

Háttérmemória: merevlemez





Háttérmemória: Solid State Drive

- A HDD-k alternatívája
- Flash-memóriát alkalmaznak bennük, amik azután megtartják az adatot, miután az áramforrás megszűnik
- Készítenek hibrid meghajtókat is, amikor a HDD-be teszik az flash-memóriát (gyorsítótár, vagy külön használható)
- A szabad blokkok száma befolyásolja a működés sebességét: minél több a szabad blokk, annál gyorsabb a meghajtó
- TRIM/UNMAP parancs: az OS jelzi a meghajtó felé, hogy mely blokkok szabadíthatóak fel későbbi írás céljára
- Samsung PM1633a: 15.36 TB

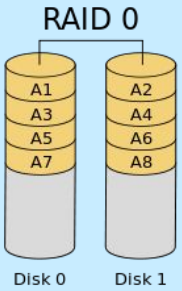
Háttérmemória: csatolófelületek

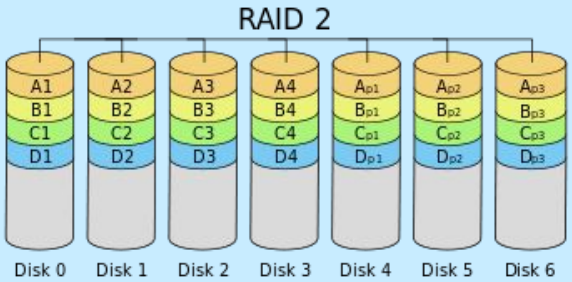
- IDE/PATA: Integrated Drive Electronics (1986)
- SCSI: Small Computer System Interface (1981)
- SATA: Serial AT Attachment (2003)
- SAS: Serial attached SCSI(2004)
- iSCSI: Internet Small Computer Systems Interface (2000)

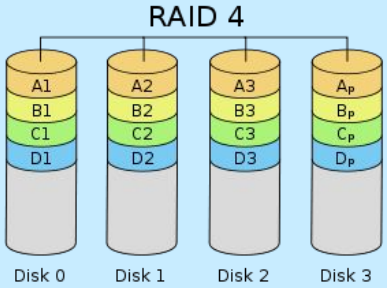
Háttérmemória: RAID

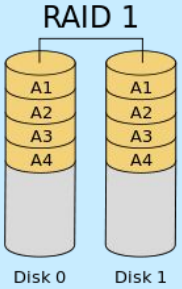
- Redundant Array of Independent Disks
- Levels
  - RAID 0: összefűzés vagy csíkozás
  - RAID 1: tükrözés
  - RAID 2: csíkozás + hibajavító kód tároló lemezek
  - RAID 3: 3-hoz hasonló, de csak paritásinfó van tárolva
  - RAID 4: 4-hez hasonló, csak nagy méretű csíkokkal
  - RAID 5: paritásinfó az összes meghajtón eloszlatva
  - RAID 6: 5 bővítés, paritás soronként és oszloponként
  - RAID 1+0: 4 lemez kell, először tükrözés, az után csíkozás
  - RAID 0+1: 4 lemez kell, először csíkozás, az után tükrözés

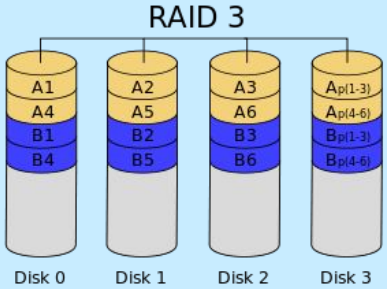
Háttérmemória: RAID

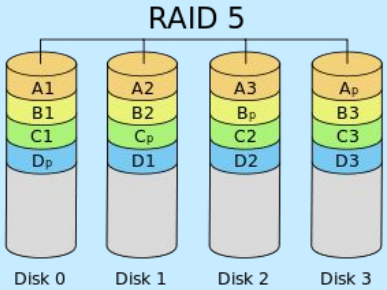


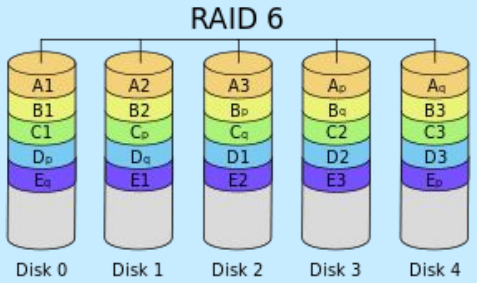












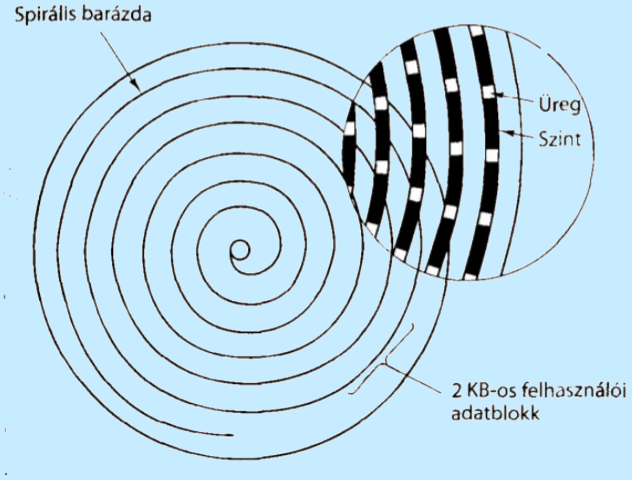
forrás: Wikipedia

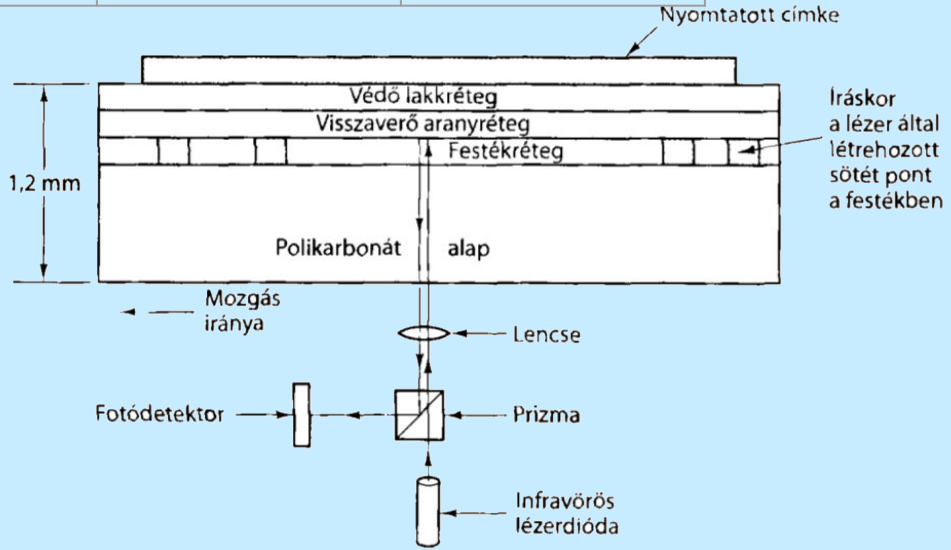
Háttérmemória: optikai meghajtók

- A műanyag lemezen tárolt adattartalmat lézerfej segítségével írja / olvassa a meghajtó
- Egy gyári CD lemez készítése: nagy energiájú IR lézerrel 0,8 mikron átmérőjű lyukakat égetnek egy bevonattal ellátott üveg mesterlemezbe, amiről negatív öntőforma készül, végül az öntőformába olvadt polikarbonátot töltenek
- A CD visszaolvasakor egy lézerdiófa 0.78 mikron hullámhosszú IR lézerrel megvilágítja a lyukakat a lemezen
- A lyukak mélysége a lézer hullámhossz negyede, ezért fáziseltolódás van a környezetről és lyukból visszavert

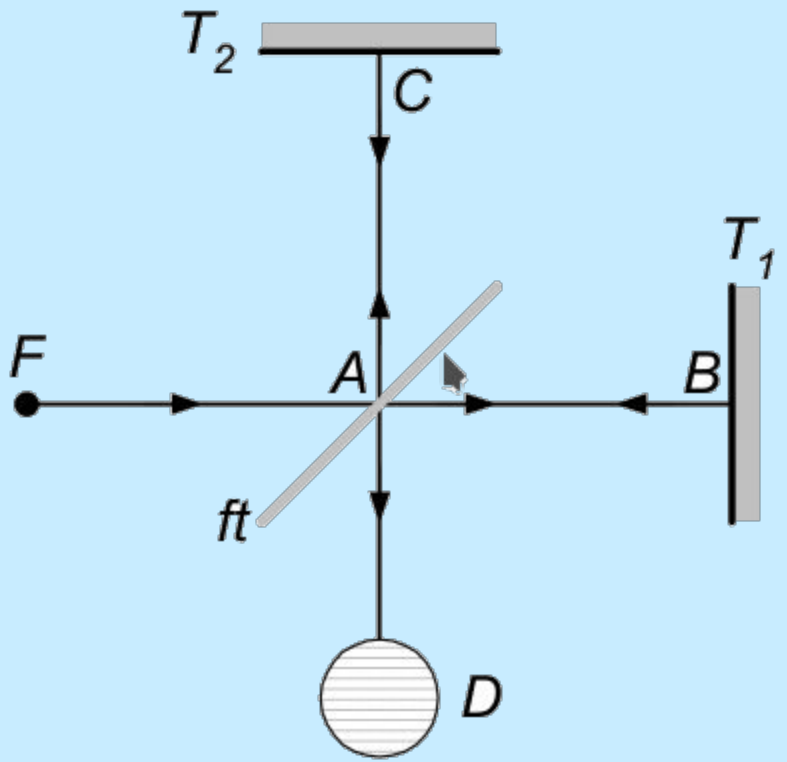
Háttérmemória: optikai meghajtók

CD	1982	700 MB	780 nm	1200 kbit/s
DVD	1995	17.08 GB	650 nm	10.5 Mbit/s
Blu-ray	2007	128 GB	405 nm	576 Mbit/s





Michelson-interferométer



Az F fényforrásból kiinduló fény az A pontban eléri a fénysugár irányával 45° szöget bezáró ft félig áteresztő tükröt.

A tükör a fényintenzitás egy részét átengedi, és ez a rész a T1 tükörről visszaverődve visszaér az A pontba, majd egy része az ft tükrön visszaverődve a detektáló eszközbe (D) jut.

A fényintenzitás másik részét az ft tükör az eredeti fénysugárra merőleges irányban visszaveri, így az a T2 tükörre kerül.

Onnan visszaverődik, és egy része az ft tükrön áthaladva a detektáló eszközbe jut.

Hullámkád:
http://www.falstad.com/ripple/

Háttérmemória: szalagos meghajtók

- Az adatok rögzítése szekvenciálisan mágnesszalagra történik
- 1951: Remington Rand - UNISERVO (224 kB)
- 2014: IBM - TS1150 (10 TB, 360 MB/s)
- “shoe-shining”: a mai gyors meghajtók puffer kifogyás esetén nem képesek azonnal megállni, vissza kell állniuk egy korábbi állapotba és újrakezdeni az írást - ha ez gyakran megesik az “fényesíti” a szalagot

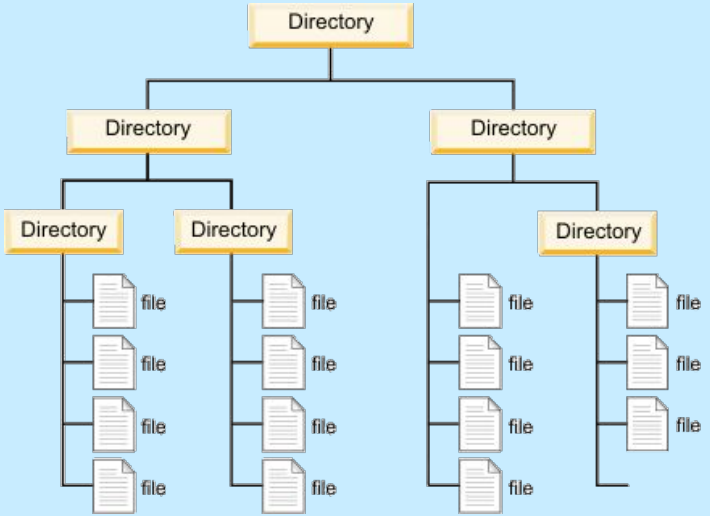
Háttérmemória: szalagos meghajtók

Remington Rand - UNISERVO

IBM - TS1150 & cardigate

Fájlrendszerek

- Problémák amik a fájlrendszerekhez vezettek:
  - A memória kicsi ahhoz, hogy minden adat elférjen benne
  - A memória illékony, a processzus végeztével nem érhető el az adat
  - Biztosítani kell, hogy egy adathoz egy időben több processus is hozzáférhessen
- Az alapegység a fájl
- A fájlok a legtöbb esetben könytárakhoz vannak rendelve, melyek fa-struktúra szerint rendezettek



Fájlrendszerek: fájlok

- Absztrakciós mechanizmus, lehetővé teszi az információ lemezen tárolását és visszaolvasását
- Fájlnév: karakterek sorozata
- Fájltípusok (Linux):
  - Könyvtárat szimbolizáló fájl
  - Normál fájl (bináris, ASCII, …)
  - Speciális fájlok
    - Block file
    - Character device file
    - Named pipe file or just a pipe file
    - Symbolic link file
    - Socket file

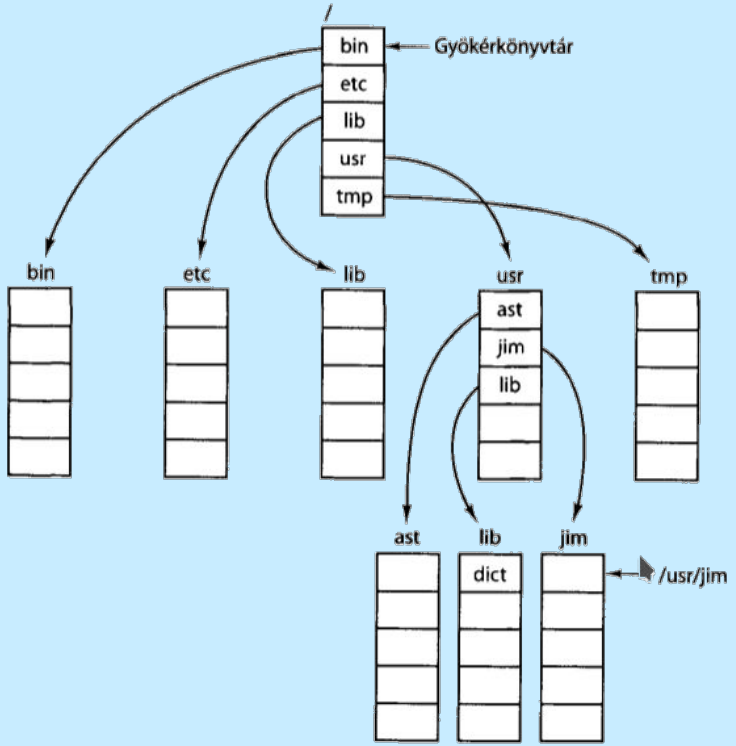
Fájlrendszerek: fájlok

- Fájlattribútumok (táblázat)
- Fájlműveletek:
  - Létesítés
  - Törlés
  - Megnyitás (írásra, olvasásra)
  - Lezárás
  - Olvasás
  - Írás
  - Hozzátoldás (append)
  - Pozícionálás (seek)
  - Attribútum írás
  - Attribútum olvasás
  - Átnevezés
  - Zárolás (lock)

Mező	Értelmezés
Védelem	Ki érheti el a fájlt és milyen módon
Jelszó	Jelszó, amelyet az eléréshez meg kell adni
Létrehozó	A fájl létrehozójának azonosítója
Tulajdonos	Az aktuális tulajdonos azonosítója
Csak olvasható jelző	0, ha írás és olvasás megengedett, 1, ha csak olvasható
Rejtettségi jelző	0 a normál eset, 1, ha listázásban nem megjelenítendő
Rendszerjelző	0 normál fájl, 1 rendszerfájl esetén
Archív jelző	0, ha archiválva volt, 1, ha archiválásra kijelölt
ASCII/bináris jelző	0, ha ASCII, 1, ha bináris a fájl
Közvetlen elérés jelző	0, ha csak szekvenciális, 1, ha közvetlen elérésű a fájl
Ideiglenességjelző	0, ha normál fájl, 1, ha törölni kell a processzus befejeződésekor
Zároltságjelző	0, ha nem zárolt, 1, ha zárolt a fájl
Rekord hossza	A bájtok száma egy rekordban
Kulcs pozíciója	A kulcs pozíciója a rekordban
Kulcs hossza	A kulcsmező hossza bájtokban
Létesítési idő	A fájl létrehozásának dátuma és időpontja
Utolsó hozzáférés ideje	Az utolsó hozzáférés dátuma és időpontja
Utolsó módosítás ideje	Az utolsó módosítás dátuma és időpontja
Aktuális méret	A bájtok száma a fájlban
Maximális méret	A lehetséges maximális fájlméret bájtban

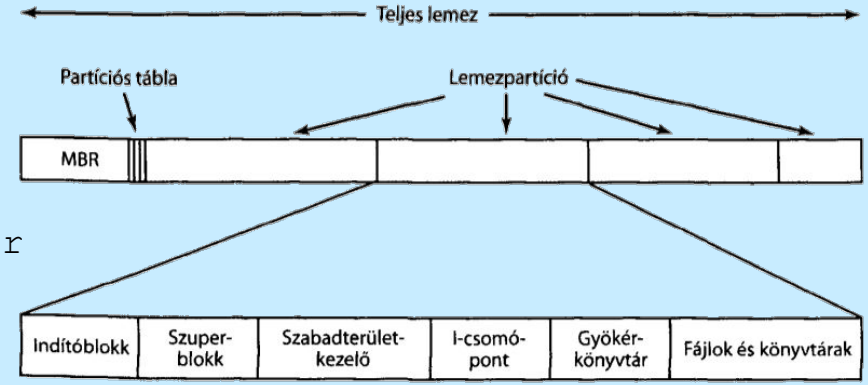
Fájlrendszerek: könyvtárak

- Könyvtárszerkezet:
  - Egyszerű
  - Hiearchikus
- Útvonal megadása:
  - Abszolút
  - Relatív
- Könyvtári műveletek:
  - Létesít, töröl
  - Megnyit, lezár
  - Olvas
  - Átnevez
  - Kapcsol, lekapcsol (link, unlink)



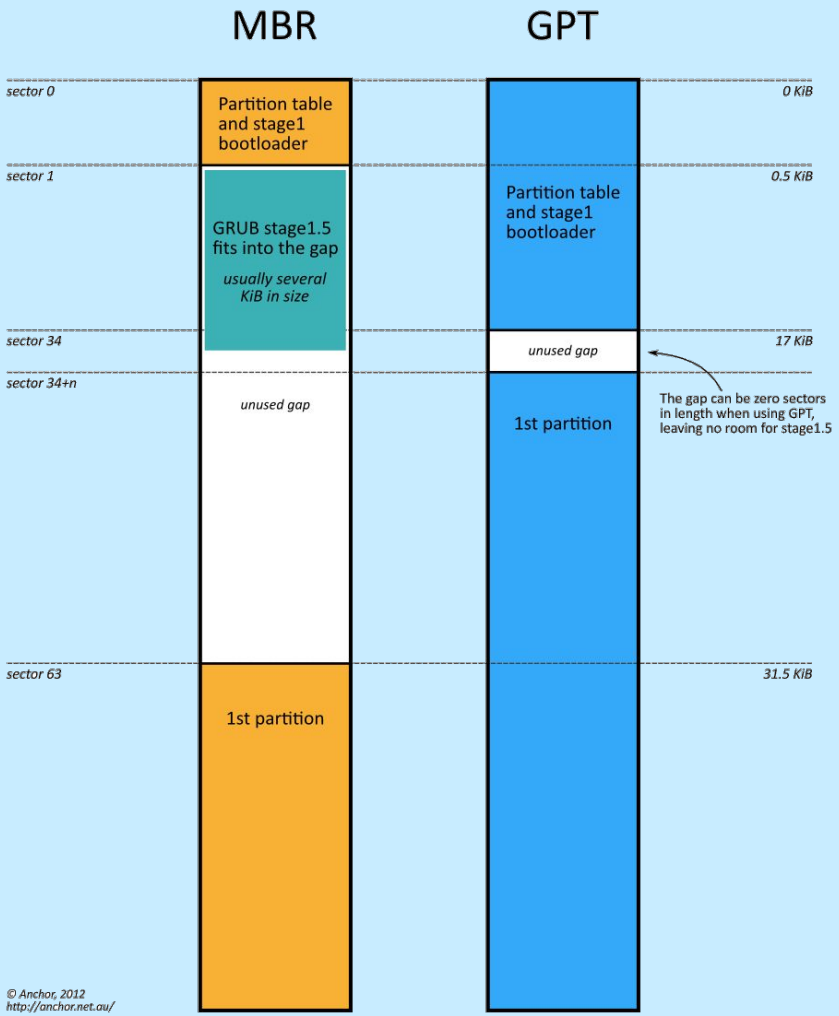
Fájlrendszerek: szerkezet

- MBR partíciós tábla
  - A lemez partíciókra van osztva
  - A lemez 0. szketora az MBR (Master Boot Record)
  - Az MBR-ben lévő kódot induláskor a BIOS tölti be
  - Az MBR után következik a partíciós tábla
  - A tábla tartalmazza, hogy a partíciók a lemezen hol helyezkednek el
  - Minden partíció független fájlrendszert tartalmaz
  - PC kompatibilis rendszerekben 4 elsődleges partíció lehet
  - Egy elsődleges partíciót definiálhatunk kiterjesztett partícióként, ami logikai partíciók láncolt listáját tartalmazhatja



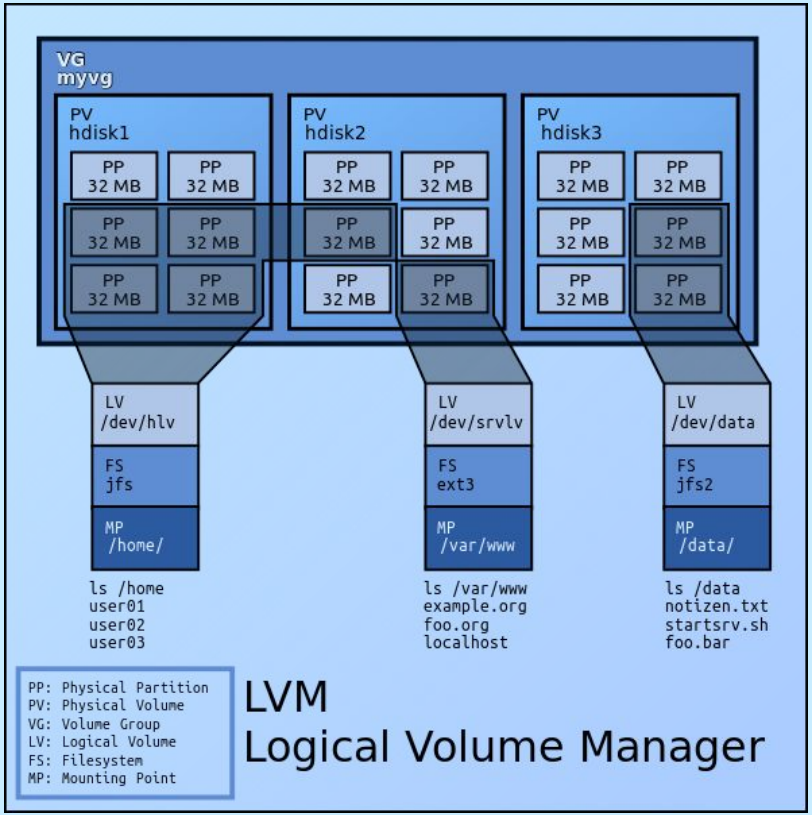
Fájlrendszerek: szerkezet

- GPT prtíciós tábla
  - GUID-t (Globally Unique Identifier) használ a lemezek és a partíciók azonosítására
  - Korlátlan számú partíció
  - 64 bit LBA -> max. 2 ZiB (Zebibyte)
  - kilo < mega < giga < tera < peta < exa < zetta < yotta
  - Backup a lemez végén
  - CRC32 ellenőrzőösszeg használata az adatsérülés detektálásához



Fájlrendszerek: Logical Volume management (LVM)

- Linux specifikus logikai kötetkezelés
- 1998-ban írta Heinz Mauelshagen a HP-UX kötetkezelője alapján
- A fizikai partíciók fölött lévő újabb absztrakciós szint
- Szintjei:
  - Physical volumes (PV)
  - Volume groups (VG)
  - Logical volumes (LV)
- Leegyszerűsíti a partíciók kezelését



Fájlrendszerek: Fontosabb PC fájlrendszerek

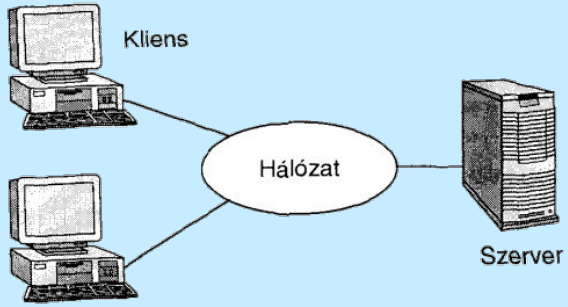
- ext2: Natív Linux FS, felfelé kompatibilis
- ext3, ext4: az ext2 naplózó verziói
- reiserfs: robosztus FS, jól kezeli az adatkorrupciót
- jfs: naplózó FS, IBM fejlesztés
- xfs: magas teljesítmény, nagy fájlok esetében is
- zfs: FS és LVM egyben, a SUN fejlesztése
- nfs: hálózati fájlrendszer
- FAT, FAT32, exFAT: Microsoft MS-DOS FS és újabb verziói
- NTFS: Microsoft legfejlettebb FS-e

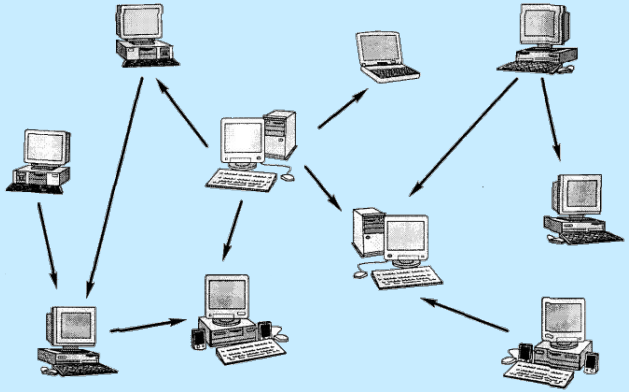
6. ÓRA
I. ZH

7. óra
Számítógépes hálózatok

Számítógép hálózatok használata

- Ajánlott irodalom: Andrew S. Tanenbaum: Számítógép hálózatok





Társadalmi vonatkozások

- A hálózatok széleskörű megjelenése új társadalmi, etikai és politikai problémákat vet fel (Laudon, 1995)
- Mi a gond a hírcsoportokkal, közösség portálokkal?
  - Amíg hobbiról, műszaki dolgokról folyik a beszélgetés nincs gond
  - Érzékeny témáknál jönnek elő a bajok, mint pl. politika, vallás, szexualitás
  - Gyakran kontrollálatlan, a vélemények megsérthetnek másokat
  - Nem csak szöveg, de kép, hang és videó anyagokat is publikálhatunk
- Kormány és állampolgár viszonya: eszközt ad a titkosszolgálatok kezébe, megfigyelhetővé válnak az emberek
- Anonim módon űzhető tevékenységek: levelezés, vásárlás-fizetés, stb.

Protokolhiearchiák

- A hálózati hardverek és szoftverek bonyolultsága miatt strukturálásra volt szükség, ez vezetett protokolhiearchiák kialakításához
- A legtöbb hálózatot úgy alakítják ki, hogy egymásra épülő rétegekre (layer) tagolják a rendszert
- Minden réteg célja, hogy szolgáltatásokat (service) nyújtson a felette és alatta lévő rétegek számára
- Protokol: az egymással kommunikáló felek (->rétegek) közötti párbeszéd szabályait tartalmazza

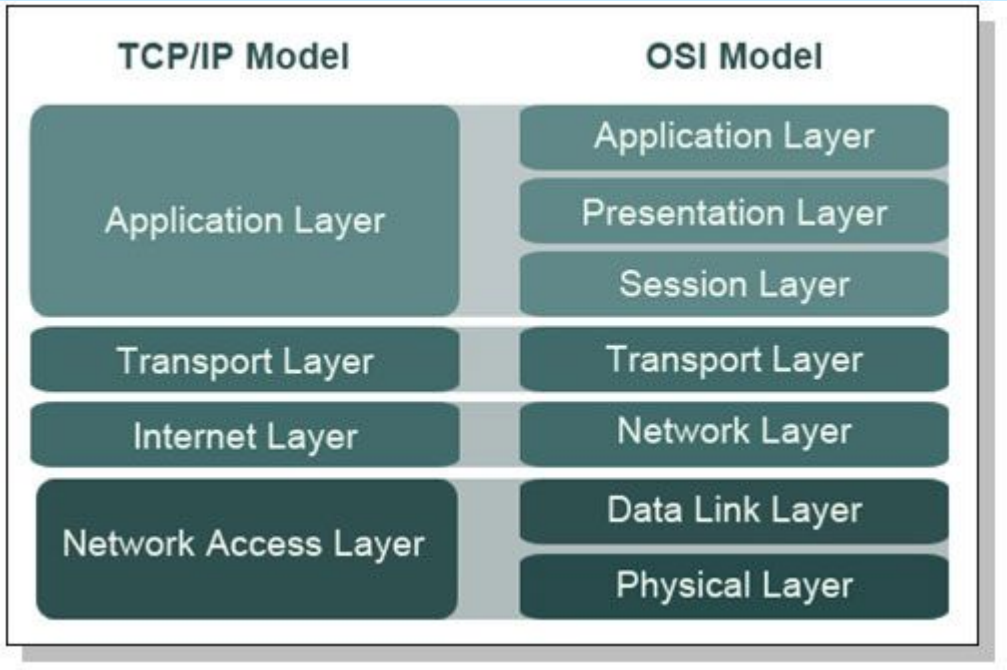
A rétegek tervezési kérdései

- Minden rétegben kell legyen egy címzési mechanizmus, ami az üzenet küldőjét és vevőjét azonosítja
- Hibavédelem: a fizikai átvitelt megvalósító áramkörök nem tökéletesek, hibajelző és hibajavító kódolásokat alkalmaznak a hibák kiszűrésére
- Forgalomszabályozás: meg kell akadályozni, hogy a gyorsabban adó gépek elárasszák a hálózatot és ezzel ellehetlenítség a lassabb adók kommunkációját
- Multiplexelés, demultiplexelés: egy összeköttetésen több párbeszédet folytató folyamatpár átvitele
- Forgalomirányítás (routing): hálózatok összekapcsolása

Az OSI hivatkozási modell

- ISO OSI: Open System Interconnection
- A különböző rétegekben használt protokolok első nemzetközi szabványosítása
- Rétegekre bontás szabályai:
  - A rétegek különböző absztrakciós szinteket képviseljenek
  - Minden réteg jól definiált feladatot hajtson végre
  - A rétegek szabványosításakor nemzetközi protokolokat kell figyelembe venni
  - Rétegek határait úgy kell meghatározni, hogy a rétegek közötti információcsere minimális legyen
  - A rétegek számát úgy kell meghatározni, hogy alapvetően eltérő feladatok ne kerüljenek egy rétegbe, de ne is váljon kezelhetetlenné a modell a rétegek nagy száma miatt

Az OSI és a TCP/IP hivatkozási modell összehasonlítása



ARPANET

- Az Internet őse, az Amerikai Védelmi Minisztérium kísérleti hálózata volt
- Alapvető szempont volt, hogy lehetővé tegye tetszőlegesen sok hálózat zökkenésmentes összekapcsolását
- Eleinte bérelt vonalakat használtak, később műholdakat és rádiós hálózatokat is hozzákapcsoltak
- Másik fontos szempont volt, hogy amíg a forrás és célállomások jól működnek, a folyó beszélgetést ne zavarja alhálózatok kiesése (decentralizált és redundáns útvonalak)
- Két legismertebb protokolja nyomán vált híressé: TCP/IP

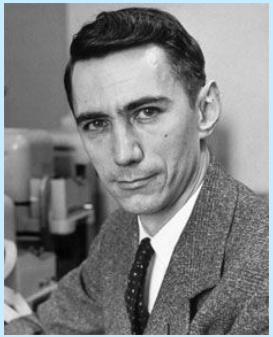
Fizikai réteg: Adatátvitel

- A 19. század elején Jean-Baptiste Fourier francia matematikus bebizonyította, hogy bármely T periódusidővel rendelkező, periodikus g(t) függvény előállítható szinuszos és koszinuszos tagok (általában végtelen) összegeként
- Az adatátvitel során energiaveszteség lép fel, ami az időtartományban változó lehet
- Sávkorlátozott jelek: A 0 és f_c vágási frekvencia között a jel kis mértékben torzul, felette arányosan nagyobb mértékben, ezért érdemes a jel frekvenciáját korlátozni

Fizikai réteg: Adatátvitel

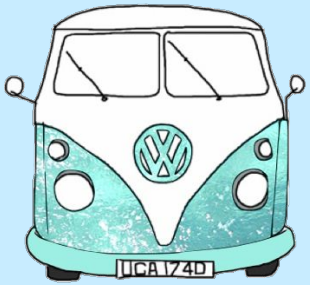
- Sávszélesség: az a frekvenciatartomány, amin belül a csillapítás mértéke “nem túl nagy”
- Nyquist-Shannon tétel: a $0 \ldots f_0$ frekvenciatartományba eső időfüggvény véges számú minta segítségével, információvesztés nélkül átvihető, ha az f_{minta} mintavételezési frekvencia f_0 frekvenciának legalább kétszerese: $f_{minta} \geq 2\ f_0$

Harry Nyquist

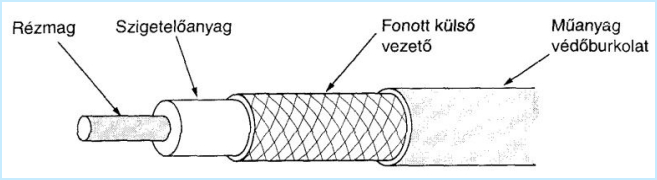


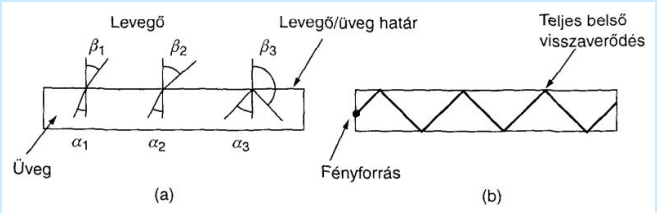
Claude Shannon

Fizikai réteg: Vezetékes átviteli közeg



- Számos átviteli módszer létezik:
  - Mágneses adathordozó (nem szabad lebecsülni annak a furgonnak az átviteli sebességét, ami mágnesszalagokkal telepakolva száguld az autópályán)
  - Sodrott érpár (twisted pair)
  - Koaxális kábel
  - Fényvezető szálak

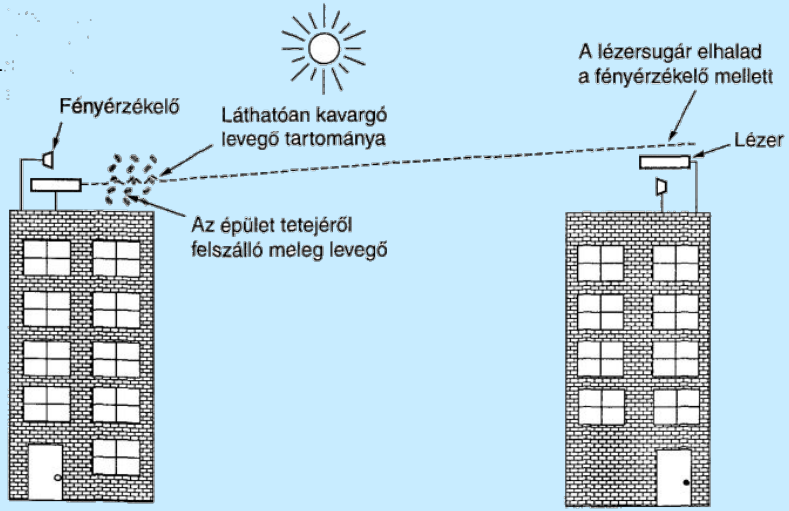




Fizikai réteg: Vezeték nélküli adatátvitel

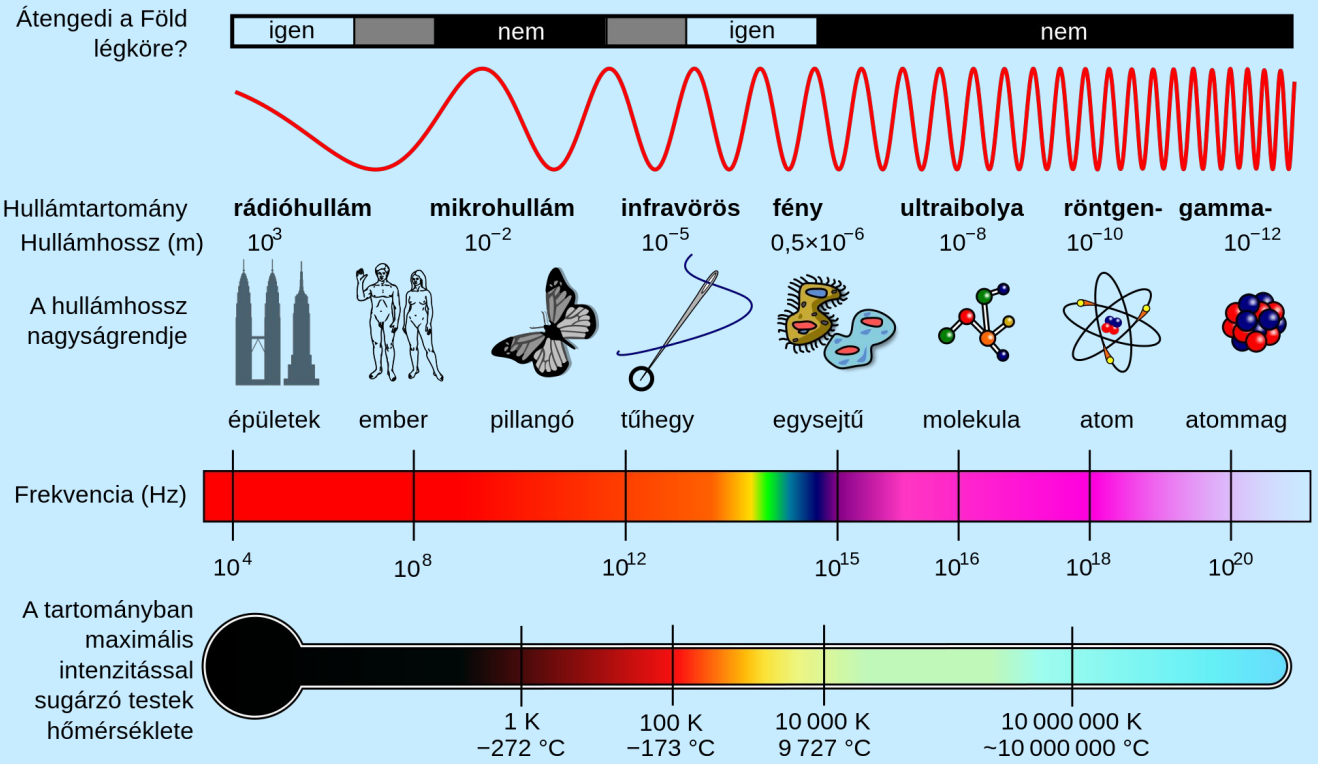
- Előnyös, ha zavaró a helyhez kötöttség, vagy, ha a vezetékes adatvitel kiépítése nehézkes lenne
- Fajtái:
  - Rádiófrekvenciás átvitel
  - Mikrohullámú átvitel
  - Infravörös és mm-es hullámú átvitel
  - Látható fényhullámú átvitel
  - Ultrahang alapú átvitel

- Az ábrán egy, a lézeres adatátvitelt befolyásoló tényező látható



Fizikai réteg: Vezeték nélküli adatátvitel

Az elektro-
mágneses hullámok
spektruma



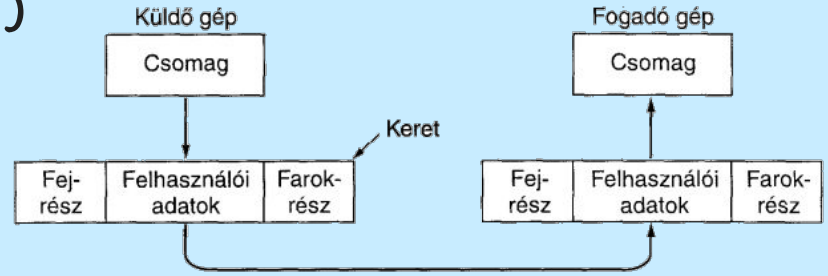
Fizikai réteg: Egyéb kommunikációs megoldások

- Kommunikációs műholdak
- Nyilvános kapcsolt telefonhálózat
- Mobiltelefon rendszer
- Kábeltelevízió (DOCSIS, ...)

Adatkapcsolati réteg: Tervezési szempontjai

- A probléma egyszerű lenne, azonban:
  - a kommunikációs áramkörök néha hibáznak
  - nem nulla késleltetéssel továbbítják az adatokat
  - véges az adatátviteli sebességük
- Megoldások:
  - Keretezés
  - Hibakezelés
  - Forgalomszabályozás

Adatkapcsolati réteg: Keretezés



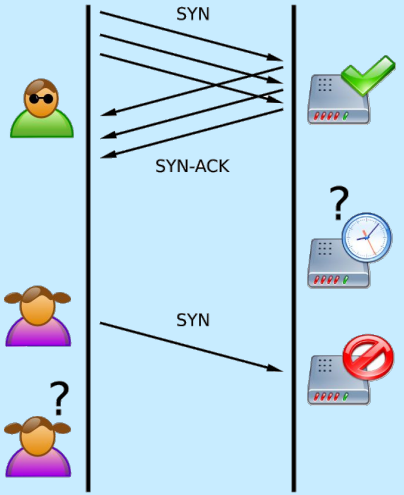
- Az érkező bitsorozat hibamentességét a fizikai réteg nem garantálja
- A bitek száma lehet kevesebb vagy több, mint az elküldötteké, és az értékük is különbözhet az eredetitől.
- Az adatkapcsolati réteg feladata, hogy jelezze, illetve - ha szükséges - kijavítsa a hibákat
- Keretézési módszerek:
  - Karakterszámlálás
  - Kezdő- és végkarakterek karakterbeszúrással
  - Kezdő- és végjelek bitbeszúrással
  - Fizikai rétegbeli kódolássértés

Adatkapcsolati réteg: Hibakezelés

- Az adónak valamilyen visszacsatolást biztosítunk arról, hogy mi történik a vonal másik végén
- Hiba esetén az adó újra-adja a keretet
- Ha egy keret elveszik, a nyugta nem érkezne meg a küldőhöz a fogadótól és így csak várna az örökkévalóságig, ezért bevezették az időzítéseket
- Ha viszont a keret vagy a nyugta elveszik, az időzítő lejár, és jelzi az adónak, hogy valószínűleg hiba történt
- Megoldás: egyszerűen újra elküldeni a kere-tet, azonban sorszámozni kell, a duplikált küldés elkerülése miatt

Adatkapcsolati réteg: Forgalomszabályozás

- Feedback-based flow control (Visszacsatolás alapú forgalom-szabályozás): A vevő információt küld az adónak, amiben engedélyezi a további csomagok küldését
- Rate-based flow control (Sebesség alapú forgalom szabályozás): A protokollba be van épít-ve egy sebességkorlát, amelyet minden küldőnek minden adattovábbítás során tiszte-letben kell tartania
- *Támadási felület: DDoS (Distributed Denial of Service) attack, vagyis elosztott szolgáltatás- megtagadással járó támadás, pl. SYN flood*



Hálózati réteg: Nyújtott szolgáltatások

- A hálózati réteg feladata, hogy a csomagokat a forrástól egészen a célig eljuttassa (az adatkapcsolati rétegnél a keretet kellett továbbítani a vonal két végpontja között)
- A hálózati rétegnek ismernie kell a kommunikációs alhá-lózat (vagyis a routerek halmaza) topológiáját, és megfelelő útvonalakat kell találnia azon keresztül
- Úgy kell kiválassza a routereket, hogy elkerülje néhány kommunikációs vonal és router túlterhelését
- Ha forrás és a cél különböző hálózatokban vannak, ezt a problémát is a hálózati réteg oldja meg

Hálózati réteg: Tervezési kérdések

- A szolgálatoknak függetleneknek kell lenniük az alhálózat kialakításától
- A szállítási réteg elől el kell takarni a jelenlevő alhálózatok számát, típusát és to-pológiáját
- A szállítási réteg rendelkezésére bocsátott hálózati címeknek egységes számozási rendszert kell alkotniuk, még LAN-ok és WAN-ok esetén is

Hálózati réteg: Forgalomirányítás

- A forgalomirányító algoritmus (routing algorithm) a hálózati réteg szoftverének azon része, amely azért a döntésért felelős, hogy egy bejövő csomag melyik kimeneti vonalon kerüljön továbbításra
- Nem adaptív, vagy statikus forgalomirányítás nem használ fel méréseket az útvonalak meghatározása során
- Az adaptív algoritmusok (adaptive algorithms) úgy változtat-ják forgalomirányítási döntéseiket, hogy tükrözzék a topológiában és rendszerint a forgalomban is történt változásokat

Hálózati réteg: az IP protokol

- A ragasztó, amely az Internetet egyben tartja
- A kezde-tektől a hálózatok összekapcsolását szem előtt tartva tervezték
- Az Interneten a kommunikáció úgy törénik, hogy a szállítási réteg az adatfolyamot datagramokra tördeli
- Egy IP-datagram egy fejrészből és egy szövegrészből áll, a fejrésznek van egy 20 bájtos rögzített része és egy változó hosszúságú opcioná-lis része
- A fejrészben található: verzió, szolgálat típusa, teljes hossz, azonsítás, élettartam, protokoll, CRC, ...

Hálózati réteg: IP-címek

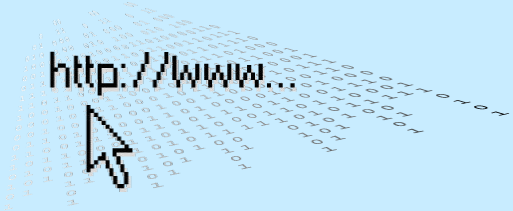
- Az Interneten minden hosztnak és routernek van egy IP-címe, amely a hálózat számát és a hoszt számát kódolja
- Jelenleg még az IPv4 az elterjedt, azonban ez kb. 2^{56} címet jelent, ami kevésnek bizonyult
- Az IPv6 32 helyett 128 bitesek, fontos jellmezői:
  - Megnövelt címtartomány
  - Közvetlen végponti címezhetőség
  - Automatikus konfiguráció
  - Hálózati mobilitás (roaminghoz hasonló)
  - Titkosítás, azonosítás (IPsec biztonsági protokoll)
  - Többszörös címezhetőség, szabványosított multicast

Szállítási réteg: nyújtott szolgáltatás

- Feladata az, hogy megbízható, gazdaságos adatszállítást biztosítson a forráshoszttól a célhosztig, függetlenül magától a fizikai hálózattól vagy az aktuálisan használt kommunikációs alhálózatoktól
- TCP (Transmission Control Protocol)
  - megbízható, összeköttetés alapú protokol, hibamentes bájtos átvitel
- UDP (User Datagram Protocol)
  - nem megbízható, összeköttetés nélküli protokol - gyors, de nem biztosított a csomagok érkezési sorrendje

Alkalmazási réteg

- Fontosabb alkalmazások:
  - Domain Name System (DNS) <- csak ezt részletezzük
  - E-mail
  - Multimédia (kép és hangátvitel)
  - Hipertext Transfer Protocol (HTTP)
  - World Wide Web (WWW)



Alkalmazási réteg: DNS

- Domain Name System (Körzeti Névkezelő Rendszer)
- Névről IP címre történő leképzés módszere
- A DNS vázlatos működése:
  - Egy felhasználói program meghívja a névvel, mint paraméterrel a címfel-oldó (resolver) nevű könyvtári eljárást
  - A címfeloldó elküld egy UDP-csomagot a he-lyi DNS-szervernek
  - A szerver megkeresi és visszaküldi az IP-címet a címfeloldónak, ami visszaadja azt a hívónak
  - Az IP-cím birtokában a program már felépítheti a TCP-kapcsolatot a célgéppel, vagy küldhet neki UDP-csomagokat
- Pl.: itk.ppke.hu -> 2001:738:5404:41::4 (IPv6)
  -> 193.224.224.4 (IPv4)

Hálózati biztonság

- A problémák nagyjából négy, szorosan össze- függő területre oszthatók:
  - titkosság (secrecy vagy confidentiality)
  - hitelesség (authentication)
  - letagadhatatlanság (nonrepudiation)
  - sértetlenség (integrity)

Hálózati biztonság: Kriptográfia

- Eredetileg négy csoport alkalmazta és vitte tovább a kriptográfia mesterségét: a had-sereg, diplomáciai testületek, naplóírók és a szerelmesek
- Titkosítás lépései:
  - A kódolandó üzeneteket (plaintext) egy olyan függvénnyel transzformálunk, melynek paramétere egy kulcs (key)
  - A titkosító eljárás kimenetét (ciphertext) továbbítjuk
  - Feltételezzük, hogy az ellenség vagy a támadó hallja és
  - pontosan lemásolhatja a teljes, titkosított szöveget
  - Ennek ellenére, az eredeti címzet-tel ellentétben, a támadó (intruder) nem ismeri a visszakódoláshoz szükséges kul-csot, és így nem képes az üzenet egyszerű visszaalakítására

Hálózati biztonság: Alapvetö kritográfiai elvek

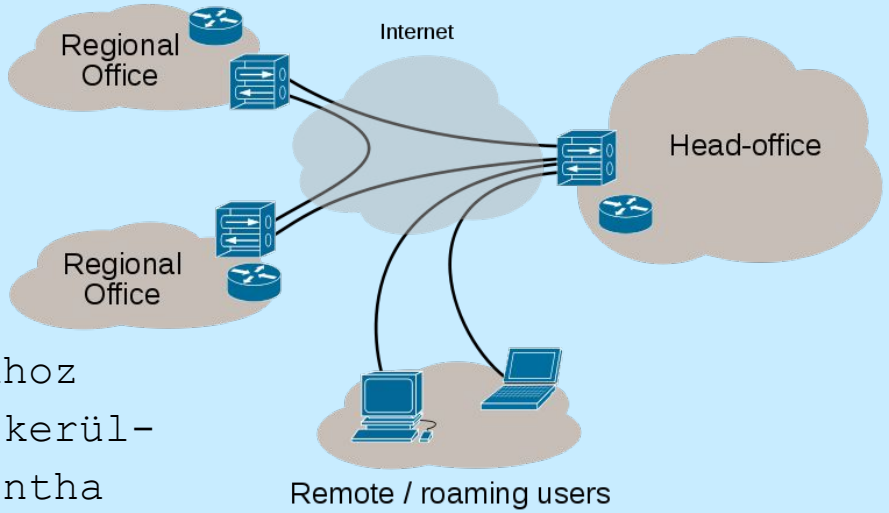
- Az üzeneteknek valamilyen redundanciát kell tartalmazniuk
  - Célja, hogy az aktív támadó ne küldhessen zagyvaságot a vevőnek
  - Sajnos a redundancia kedvez a passzív támadóknak, ezért nem szabad a redundanciát az üzenet eleján nullák sorozatéval elérni
- Kell egy módszer az ismétléses támadások meghiú-sítására
  - Meg kell oldani, hogy ellenzőrizhető legyen az üzenetek frissessége
  - Elérhető pl. időbélyeg alkalmazásával

Hálózati biztonság: RSA

- Nyilvános kulcsú titkosítási módszer
- A számelmélet tételein alapszik:
  - Válasszunk két nagy (jellemzően 1024 bites) prímszámot, p-t és q-t
  - Számoljuk ki az n = p * q és az z = (p - 1) * (q - 1) számokat
  - Válasszunk egy z-hez relatív prímet, jelöljük d-vel.
  - Keressünk egy olyan e számot, melyre e * d = 1 mod z
- A módszer biztonsága a nagy számok faktorizálásának nehézségén alapszik

Hálózati biztonság: VPN

- Virtual Private Network, vagyis Virtuális Magánhálózat
- A számítógépes hálózat fölé van építve
- A VPN-be bejelentkezett gép ahhoz hasonló közvetlen kapcsolatba kerülhet a VPN többi kliensével, mintha valóban egy alhálózat tagjai lennének
- Mivel a kapcsolat titkosított, így a hálózat privát



Hálózati biztonság: Tüzfalak

- A tűzfal (firewall) célja a számítástechnikában annak biztosítása, hogy a hálózaton keresztül egy adott számítógépbe ne történhessen illetéktelen behatolás
- Szoftver- és hardverkomponensekből áll
- Hardverkomponensei olyan hálózatfelosztó eszközök, mint a router vagy a proxy
- A szoftverkomponensek ezeknek az alkalmazási rendszerei tűzfalszoftverekkel, beleértve ezek csomag- vagy proxyszűrőit is

8. óra
A UNIX shell I.

Csatlakozás ssh-val

- A shell használatát a **balna.itk.ppke.hu** gépen fogjuk gyakorolni
- A Bourne-Again shell-t (bash) fogjuk használni
- A balna-hoz SSH-val kapcsolódunk
- Mindenkinek a felhasználóneve a NEPTUN azonosítója és a kedzeti jelszava “panda”
- Windowsról a putty SSH kliens segítségével kapcsolódunk
- Linuxról és Mac-ről pedig az ssh paraccsal
  - ssh neptunazonosito@balna.itk.ppke.hu
- Az első belépés után meg kell változtatni a jelszót:
  - $ passwd [enter]
  - a jelszót 2x kell beírni, beírás közben a kurzor nem lépked

Milyen gépet használunk?

- uname -a
  - A kernel tulajdonásgai
- free -h
  - Szabad memória ellenőrzése
- cat /proc/cpuinfo
  - A CPU(k) tulajdonságai
- lspci
  - A PCI sínhez csatlakoztatott eszközök
- lsusb
  - Az USB buszra csatlakoztatott eszközök

Segítég, hibakeresés

- man [parancs]
  - manual, információt ad a megadott parancsról
- which [parnacs]
  - amennyiben a parancs elérhető a PATH-ben megadott könyvtárak valamelyikében, kiírja, hogy mi a parancs pontos elérési útvon
- history
  - parancssori előzmények kiíratása
- dmesg
  - rendszerüzenetek kiíratása

Jelölések

- A parancsok után általában írhatjuk a --help kapcsolót, aminek hatására segítséget ad a használathoz
- A help-ben meg van adva, hogy miként paraméterezhetjük a parancsot, ezek a parancssori argumentumok
- A parancssori argumentumok esetében ami [ ] között vannak, azokat nem kötelező megadni
- Ha az argumentum kötelező, de több lehetséges opció van, akkor azt pipe-ok választják el: (op1 | op2 | op3 | ...)

Állománykezelés

- pwd
  - print working directory
  - kiírja az aktuális könyvtár abszolút elérési útját
- cd [cél könyvtár]
  - a megadott könyvtárba lép, ha az létezik
  - paraméter nékül a felhasználó saját könyvtárába lép (/home/username vagy ~)
- ls
  - kiírja az aktuális könyvtár tartalmát
  - -a: all, a rejtett fájlokat is listázza (ponttal kezdődik a nevük)
  - -l: long format, nem csak a filenevet írja ki
- mkdir új-könyvtár-neve
  - létrehoz egy új könyvtárat
- touch új-fájl-neve
  - módosítja a fájl utolsó elérésének időpontját
- cp eredeti-fájl új-fájl
  - copy: fájl másolása
  - -r kapcsolóval rekurzívan másol (könyvtárakat is)

Állománykezelés

- mv eredeti-fájl új-fájl
  - move: fájl áthelyzése / átnevezése
- rm fájl-neve
  - remove: fájl törlése
  - -r kapcsolóval rekurzív
- cat filenév
  - fájl tartalmának kiírása
- ln
  - link létrehozása
  - paraméter nelkül hardlink
  - -s: softlink
- echo “Hello World!”
  - Kiírja a megadott szöveget
- diff fájl-1 fájl-2
  - két fájl tartalmának összehasonítása
- file filename
  - kiírja, hogy a filename nevű fájlnak mi a típusa
- mc
  - Norton Commanderhez hasonlít
- lsof
  - kilistázza az aktuálisan megnyitva lévő fájlokat
- find
  - rekurzívan kilistázza az aktuális könytár fájljait és könyvtárait

Állománykezelés

- chmod filenév
  - fájl jogosultságok megvált.
  - -R: rekurzív
  - r: read, w: write, x: execute
  - Az rwx megfelel egy bináris számnak
  - Pl. r-x = 4 + 0 + 1 = 5
  - Külön beállíthatjuk a jogokat a tulajdonosra: (u)ser, a csoporttagra: (g)roup és mindenki másnak (o)ther
  - g+w : írási jog adása a csoporttag számára
    uo-rx : olvasási és futtatási jog törlése a tulajdonos és az other számára
- chown filenév
  - fájl tulajdonos / csoport megváltoztatása
  - -R: rekurzív
- sync
  - a még lemezre nem írt adatokat azonnal lemezre írja (pl. késleltetett írás esetén)
- cowsay, cowthink

```
  ^__^
 (oo)\_______
 (__)\       )\/\
     ||----w |
     ||     ||
```

9. óra
A UNIX shell II.

String müveletek I.

- | (pipe)
  - egy program kimenetének (standard output vagy standard error) továbbirányítása a pipe után lévő program standard inputjára
- > filenév
  - az program a kimenetét egy új fájlba irányítja
- >> filename
  - hozzáfűz a megadott filehoz
- grep keyword filenév
  - kiszűri azokat a sorokat, amelyek nem tartlmazzák a keyword-öt
  - -r: rekurzív, almappákben keres
  - -i: kis-nagybetű figyelmen kívül hagyása
  - -v: inverz módon működik
- head n
  - csak az első n sort engedi tovább
- tail n
  - csak az utolsó n sort engedi tovább
- less
  - fájl nézegető (kilépés q-val)
- sort
  - ABC szerint sorba rendezve engedi tovább az inputját
  - -u: unique (kiszűri az ismetlődéseket)
- wc
  - word count: sor, szó, karakter megszámolása

String műveletek II.

- `cut`
- `tr`

Rendszeradminisztráció

- du [könyvtár]
  - elfoglalt terület számítása
  - -h: human readable
  - -s: sum
  - --max-depth=n (maximális mélység)
- df [könyvtár]
  - szabad terület számítása partíciókként
  - -h: human readable
- who
  - bejelentkezett felhasználók listázása
- whoami
  - aktuális felhasználó username kiírása
- useradd username
  - felhasználó hozzáadása
  - /etc/passwd
- adduser username [groupname]
  - felhasználó hozzáadása interaktív módon
  - felhasználó hozzáadása egy csoporthoz
- groups [username]
  - felhasználó csoportjainak listázása
- addgroup groupname
  - csoport létrehozása
  - /etc/group
- passwd [user]
  - jelszó megváltoztatása

Folyamatok

- w
  - ki van belépve és mit csinál
- parancs &
  - parancs futtatása a háttérben
- [ctrl+Z]
  - parancs megállítása
- bg
  - parancs háttérbe küldése
- fg
  - parancs előtérbe hozása
- ps
  - futó folyamatok listázása
  - -ef: minden folyamat, full format
  - aux: -ef hez hasonló, kicsit bővebb
- kill PID
  - folyamat terminálása
  - -n: kill-signal megadása számmal
- killall parancs
  - terminalja a megadott nevű parancsot
  - -n: kill-signal megadása
- top
  - interaktív folyamat lista
- htop
  - színes interaktív folyamat lista

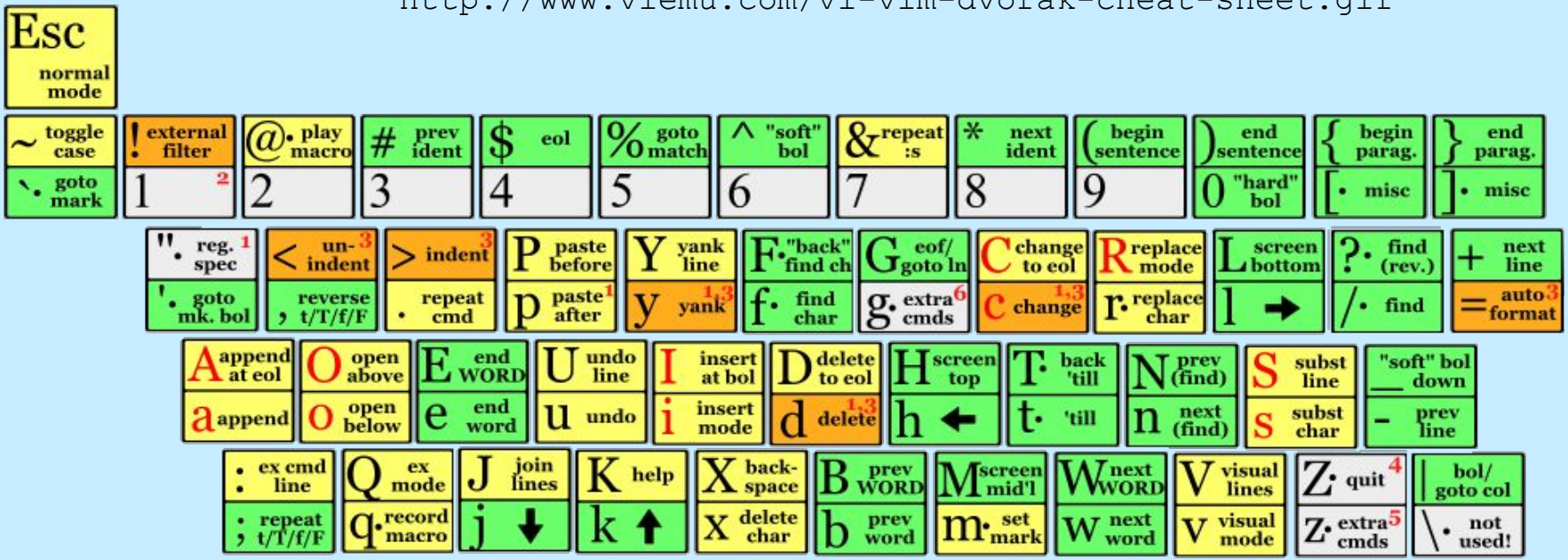
VI / VIM

- Előnye, hogy minden rendszeren telepítve van -> ha ismered, biztosan tudsz UNIX/Linux rendszerken fájlokat szerkeszteni
- Futtatás: vi [path/filename]
- Esc: parancs mód
  - vannak parancs-billentyűk
  - vagy kettőspont után írhatjuk a parancsokat
- :q -> kilépés
- :q! -> kilépés a változások mentése nélkül
- :w -> mentés (write)
- :x = :wq
- :n -> ugárs az n. sorra
- gg -> ugrás a fájl elejére
- G -> ugrás a fájl végére
- i -> insert mód (=gépelés)
- dd -> sor törlése
- d5d -> 5 sor törlése
- yy -> copy
- p -> paste
- A -> insertmód a sor végén
- r -> 1 karakter felülírása
- R -> felülíró mód
- o -> új sor a mostani után
- O -> új sor a mostani előtt
- u -> undo
- U -> minden változás visszavonása

VI / VIM

Cheat-sheet

http://www.viemu.com/vi-vim-dvorak-cheat-sheet.gif



A shell változók használata


```
$ FOO=”ASD”
```

- A változó teljes nevét NAGYBETŰKKEL írjuk, így könnyen megkülönböztethetőek
- Változó deklarálása: VÁLTOZONÉV=”a változó tartalma”

```
$ echo $FOO
ASD
```

- A változóra a $ karakterrel hivatkozunk, a parancsértelmező a $VÁLTOZTÓNÉV helyére a változó értéket helyettesíti be

```
$ unset FOO
$ FOO=””
```

- Az unset paranccsal törölhetjük a változót
- Vagy egyszerűen üres értékre állítjuk be

A shell változók használata

```
$ export FOO=”ASD”
$ BAR=”QWE”
$ bash       # subprocessz indítása
$ echo $FOO  # a FOO látható
ASD
$ echo $BAR  # a BAR értéke üres
$ exit       # kilépés a
subprocesszből
$
```

- Az export paranccsal a változót elérhetővé tesszük az adott shell-ben futtatott subprocesszek számára is.

A shell változók használata

```
$ FOO[0]=”one”
$ FOO[1]=”two”


$ BAR=(three four five)


$ echo $FOO[1]
two
$ echo ${FOO[*]}
one two

$ echo ${BAR[*]}
three four five
```

- A szöglestes zárójellel tömböt dekralálhatunk, a [és] között adjuk meg, hogy a tömb hanyadik elemére hivatkozunk
- bash-ban normál zárójelek között, szóközös felsorolással is megadhatjuk a tömb elemeit
- Ha egy elemre annak sorszámával hivatkozhatunk
- Valamennyi elemre hivatkozhatunk a * karakterrel, de ekkor a $ hatáskörét ki kell terjeszteni az egész kifejezésre kapcsos zárójelekkel: { és }

A shell változók használata

```
$ WIDTH=10
$ HEIGHT=20
$ echo Area: $((WIDTH*HEIGHT))
Area: 200
$ echo District:
$((2*WIDTH+2*HEIGHT))
District: 60

$ N=10
$ N=$((N+1))
$ ((N=N+1))
$ ((N+=1))
$ ((N++))
$ let “N = N + 1”
$ N=`expr N + 1`
$ echo $N
17
```

- Amennyiben a változóban számot tároltunk el végezhetünk vele

- Egy válztozóban tárolt szám értékét több módon is megváltoztathatjuk

A shell változók használata

```
$ X=11
$ Y=5
$ echo $((X/Y))
$ 2
$ echo $((X%Y))
$ 1

$ PI=3.14
$ R=2
$ AREA=`echo “$R*$R*$PI” | bc -l`
echo $AREA
12.56
```

- A bash csak egész számokat kezel, osztás esetén csak az egész részt írja ki
- A % használatával kapjuk meg egy adott osztás maradékát

- A ` ` (AltGr+7) jelek között megadott részt parancsként értelmezi a bash
- Ha tört számokkal szeretnénk számolni a bc parancsot használhatjuk
- Az AREA változó étéke legyen a ` ` közötti parancs eredménye, amiben a törtet tartalmazó változókon kijelölt műveletet a bc parancsra irányítjuk.

Speciális változók

`$BASH_VERSION`	Bash verzió
`$PATH`	File keresési útvonalak
`$UID`	User ID
`$GROUPS`	User csoportjai
`$HOME`	User home könyvtára
`$HOSTNAME`	Hostname
`$PWD`	Print working directory
`$PS1`	Elsődleges prompt
`$0 $1 $2 ...`	Pozicionális paraméterek
`$#`	Paraméterek száma
`$*`	Összes paraméter egy string-ként
`$@`	Összes paraméter különálló stringekként
`$!`	Az utolsó háttér process PID-je
`$?`	Az utolsó lefutott process visszatérési értéke
`$RANDOM`	Véletlen számot ad vissza 0 és 32767 között
`# set -o`	Bash opciók kilistázása

A shell-script felépítése

```
#!/bin/bash


# Name: schell script skeleton
# Purpose: education scripting basics
# Create date: 2016.11.05
# Modify date: 2016.11.05
# Created by: aphex


echo “This is a very simple script.”

exit 0
```

- Az első sorban megadjuk a parancsértelmezőt
- Ezután következik a header, amiben atokat adhatunk meg a scriptre vonatkozóan
  A # utáni rész komment, a parancsértelmező nem értelmezi
- Program
- Visszatérés a program végén 0 hibakóddal, ami arra utal, hogy a script hiba nélkül futott le

10. óra
A UNIX shell III.

Feltételes kifejezések: if

```
if [ condition ]
then
   commands
fi
-----------------------------
if [ condition ]; then
   commands
elif [ condition2 ]; then
   commands
else
   commands
fi
-----------------------------
if [[ condition ]]; then
   commands
fi
```

- Szintaxis

- A dupla szögletes zárójel használata “biztonságosabb”, több funkció érhető el

Feltételes kifejezések: tesztek

`[ -e filename-dirname ]`	Az állomány létezik
`[ -f filename ]`	Az állomány létezik és fájl
`[ -d directoryname ]`	Az állomány létezik és könyvtár
`[ -h symlink ]`	Az állomány létezik és szimbolikus link
`[ -r filename ]`	Az állomány létezik és olvasható
`[ -w filename ]`	Az állomány létezik és írható
`[ -x filename ]`	Az állomány létezik és futtatható
`[ -s filename ]`	Az állomány létezik és a mérete > 0
`[ -v varname ]`	A változó létezik
`[ -z string ]`	A string mérete 0
`[ string1 == string2 ]`	A két string megegyezik
`[ string1 != string2 ]`	A két string nem egyezik meg
`[ string1 < string2 ]`	string1 előbb van ABC sorrendben
`[ num1 OP num2 ]` `OP: -eq, -ne, -gt, -lt, -ge, -le`	Aritmetikai összehasonlítás equal, not-equal, greater-than, less-than, greater-or-equal, less-or-equal

Feltételes kifejezések: tesztek

```
[ ! expression ]
[ expr1 -a expr2 ]
[ expr1 ] && [ expr2 ]
[ expr1 -o expr2 ]
[ expr1 ] || [ expr2 ]
(( num ))
(( num1 OP num2 ))
  OP: <, >, ==, <=, >=
```

- A kifejezés ellentéte igaz
- ÉS kapcsolat expr1 és expr2 között
- ÉS kapcsolat expr1 és expr2 között
- VAGY kapcsolat expr1 és expr2 között
- VAGY kapcsolat expr1 és expr2 között
- 1 operandusú aritmetikai teszt
- 1 operandusú aritmetikai tesztek

Vezérlési szerkezetek: for

```
#!/bin/bash

# Filename: script.sh

for N in 1 3; do
    echo $N
done

exit 0

----------------------------------------
$ ./script.sh
1
3
```

- A for szerkezetben az N változó minden ciklus alkalmával az “in” után megadott értékeket veszi fel sorban
- Számos módszer létezik az “in” után következő szekvencia megadására, ahogyan a következő diákon láthatjuk majd

Vezérlési szerkezetek: for

```
#!/bin/bash

# Filename: script.sh

for N in {1..3}; do
    echo $N
done

exit 0

--------------------------------------
$ ./script.sh
1
2
3
```

- Az “in” után megadható tartomány is kapcsos zárójelek között
- {1..3} -> 1,2,3
- {1..10..2} -> 1,3,5,7,9 ekkor {START..END..INCREMENT} alakban a lépés köz is megadható

Vezérlési szerkezetek: for

```
#!/bin/bash

# Filename: script.sh

ARRAY=(one two three)
for N in `echo ${ARRAY[*]}`; do
        echo $N
done

exit 0

---------------------------------------
$ ./script.sh
one
two
three
```

- Az “in” után kifejezést is megadhatunk a már ismert ` ` jelek között

Vezérlési szerkezetek: while

```
while [ expression ]
do
    commands
done
```

- Szimpla while ciklus

```
while [ expr1 ]; do
    while [ expr2 ]; do
        commands
    done
done
```

- Halmozott while ciklus

```
while true; do
    echo This loop never ends…
done
```

- Végtelen loop

Vezérlési szerkezetek: until

```
A=0

until [ $A -gt 5 ]; do
  ((A++))
  echo $A
done

---

Kimenet:
1
2
3
4
5
6
```

- Mindaddig fut a ciklus, amíg nem igaz a feltétel

A break és a continue

```
for N in {1..3}; do
    if (($N == 2)); then
        break;
    fi
    echo $N;
done
```

- Eredmény:
  1

```
for N in {1..3}; do
    if (($N == 2)); then
        continue;
    fi;
    echo $N;
done
```

- Eredmény:
  1
  3

Függvények használata

```
#!/bin/bash

test ()
{
    echo $1
}

test “Hello World!!!”


--------------------------------

$ ./helloworld.sh
Hello World!!!
```

- A parancsértelmező megadása
- Függvény fejléc
- Függvény törzse
- Függvény meghívása
- Futtatás

Órai programok #1

```
aphex@balna:~/otodik$ cat if.sh
#!/bin/bash


if [ 10 -eq 10 ]; then
  echo "10 egyenlo 10-zel";
fi

exit 0
aphex@balna:~/otodik$ ./if.sh
10 egyenlo 10-zel
```

Órai programok #2

```
aphex@balna:~/otodik$ cat if2.sh
#!/bin/bash

if [ -f beka ]; then
  echo "BREKK!";
elif [ -f kutya ]; then
  echo "VAU!";
else
  echo "NINCS BREKK, NINCS VAU!";
fi
exit 0

aphex@balna:~/otodik$ ./if2.sh
NINCS BREKK, NINCS VAU!
```

Órai programok #3

```
aphex@balna:~/otodik$ cat for.sh
#!/bin/bash


for X in 2 4 7 8; do
  echo $X
done

exit 0
aphex@balna:~/otodik$ ./for.sh
2
4
7
8
```

Órai programok #4

```
aphex@balna:~/otodik$ cat for2.sh
#!/bin/bash

for X in {10..100..20}; do
  echo $X
done

exit 0
aphex@balna:~/otodik$ ./for2.sh
10
30
50
70
90
```

Órai programok #5

```
aphex@balna:~/otodik$ cat while.sh
#!/bin/bash
N=0
while [ $N -le 10 ]; do
  ((N=N+1))
  ((N++))
  if [ $N -eq 4 ]; then
    continue;
  fi
  echo $N
  sleep 0.5

  if [ $N -eq 6 ]; then
    break;
  fi
done
```

```
aphex@balna:~/otodik$ ./while.sh
2
6
```

10. óra
LaTeX I.

Bevezetés

- Mi a LaTeX?
  - Szövegformázó rendszer
  - Leslie Lamport alkotta meg
  - Szakdolgozatok, tudományos cikkekhez ideális
  - Része egy szövegjelölő nyelv, amiben a szöveget megjelenési információkkal láthatjuk el
  - A forrásból fordító program készít dokumentumot (PS,PDF,..)
- Offline eszközök
  - MiKTeX
  - Texmaker
- Online eszközök
  - overleaf.com <- orán ezt használjuk
- Előnyei
  - Nyomdai minőség
  - Logikai struktúrát kell leírni, a megjelenítést nem
  - Jól kezeli a képleteket
  - Kevés memóriát igényel
  - Hordozható, ingyenes
  - Nyílt forráskódú
- Hátrányai
  - Több hozzáértést igényel mint egy WYSWYG szövegszerkesztő
  - Parancssoros kezelőfelület
  - Nehéz új kinézeti tervez készíteni

Hello world

```
\documentclass{article}

\begin{document}
Hello world!
\end{document}
```

Hello world!

- A documentclass: article, book, report, slides, letter, stb. [...]: további opciókat adhatunk meg (opcionális paraméterek).
- A preambulum: a \documentclass{...} és a \begin{document} közötti rész. Az egész dokumentumra vonatkozó parancsok.
- A dokumentum teste: a \begin{document} és az \end{document} közötti rész.
- \-rel kezdve adhatunk meg parancsokat. \parancs[opcionális argumentum]{kötelező argumentum}

Ékezetes szöveg

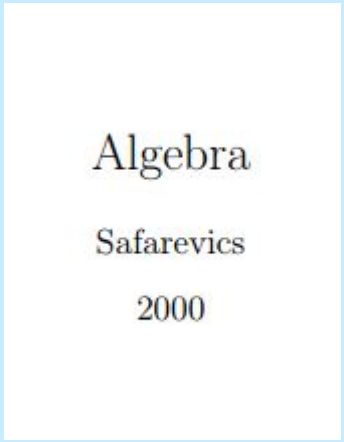
```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage{t1enc}
\usepackage[hungarian]{babel}
\selectlanguage{hungarian}
\begin{document}
%Ékezetes betűk próbája
öüóőúéáűí
\end{document}
```

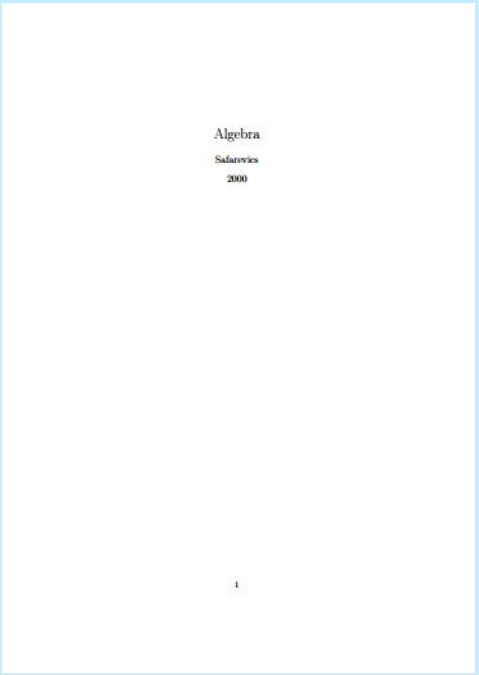
öüóőúéáűí

- `\usepackage{}`: a LaTeX képességeit programcsomagokkal bővíthetjük.
- `%`: komment, a LaTeX fordító számára láthatatlan, saját belső megjegyzések.

Címlap

```
\documentclass{article}
\author{Safarevics}
\title{Algebra}
\date{2000}

\begin{document}
\maketitle
\end{document}
```

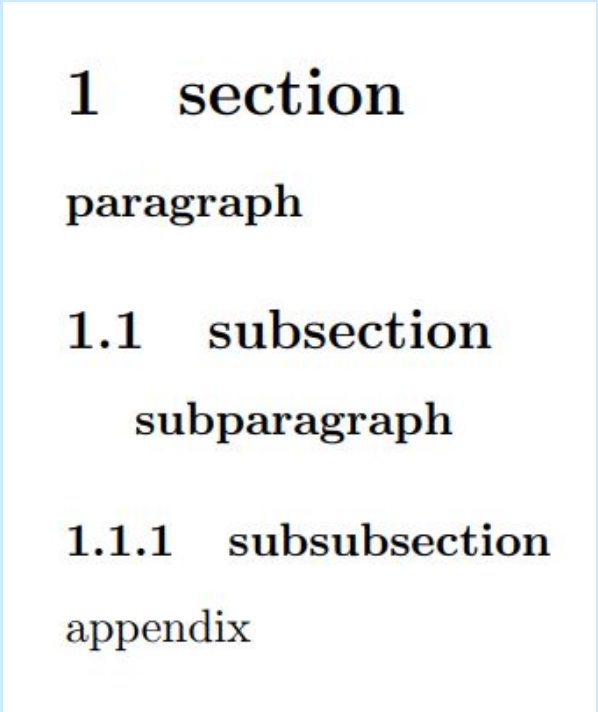




Fejezetek

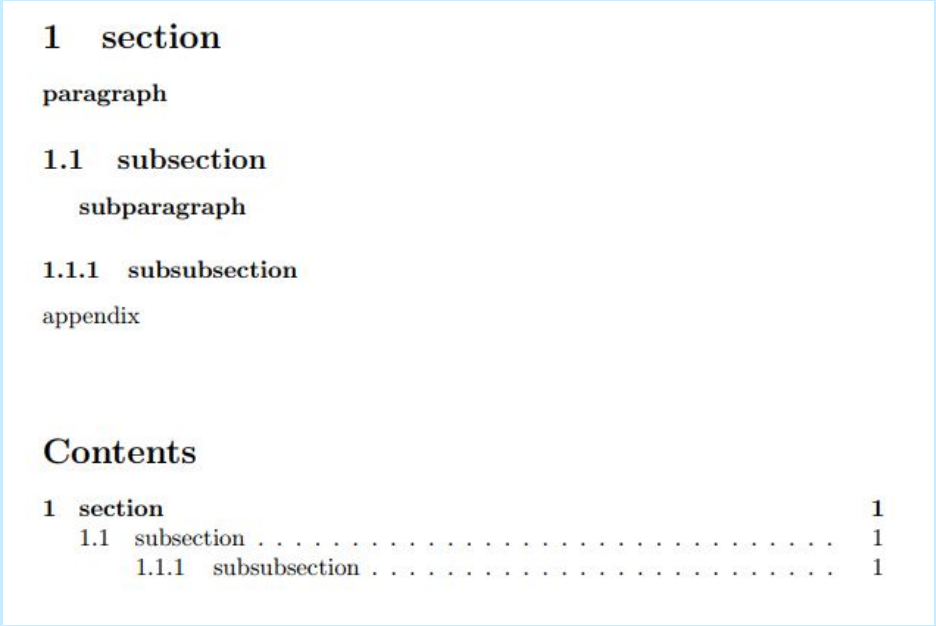
```
\section{section}
\paragraph{paragraph}
\subsection{subsection}
\subparagraph{subparagraph}
\subsubsection{subsubsection}
\appendix{appendix}
```

- Fejezetek, alfejezetek
- Paraméter: a szakasz címe, pl.: \section{Bevezetés}
- Számozás kihagyása: \section*{Bevezetés}



Tartalomjegyzék

```
\section{section}
\paragraph{paragraph}
\subsection{subsection}
\subparagraph{subparagraph}
\subsubsection{subsubsection}
\appendix{appendix}
\\
\\
\tableofcontents
```



1 section

paragraph

1.1 subsection

subparagraph

1.1.1 subsubsection

appendix

Contents

- Tartalomjegyzék

Szóközök, bekezdések

- Whitespace: mindig csak egy
- Új bekezdés: üres sor
- Non-breaking space: ~
  (a dokumentum nem tördelhető ennél a space-nél)
- Hosszú kötőjel (gondolatjel): --
- Új sor: \newline vagy \\
- Üres hely (láthatatlan szöveg)
  (valami-nyi helyet hagy üresen)

Betűváltozatok, betűméret, kiemelés

```
\textup{álló}

\textit{kurzív}

\textsc{kiskapitális}

\textmd{normál}

\textbf{félkövér}

\texttt{írógép}
```

álló

kurzív

KISKAPITÁLIS

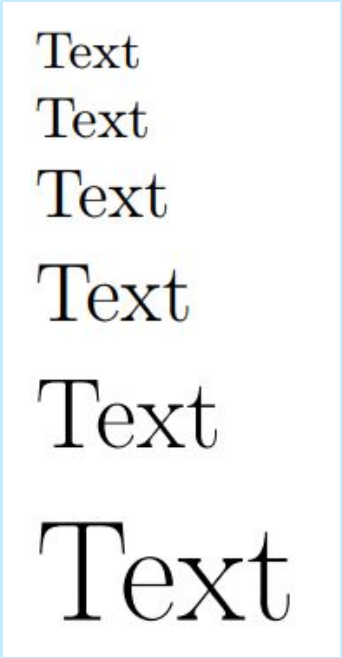
normál

félkövér

írógép

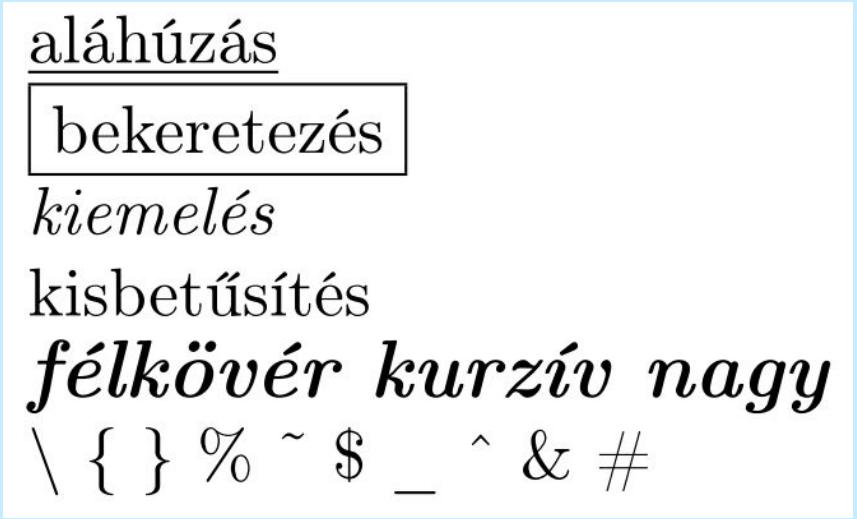
Betüváltozatok, betüméret, kiemelés

```
\small{Text}

\normalsize{Text}

\large{Text}

\Large{Text}

\LARGE{Text}

\Huge{Text}
```



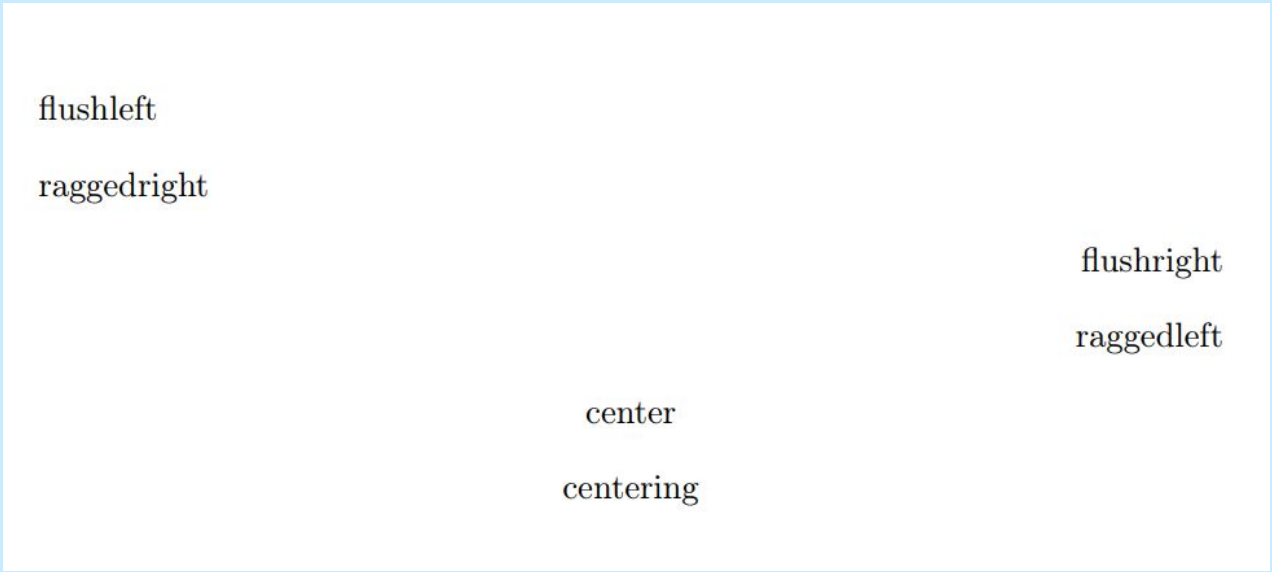
Betűváltozatok, betűméret, kiemelés, speciális karakterek

```
\underline{aláhúzás}

\framebox{bekeretezés}

\emph{kiemelés}

\MakeLowercase{KISBETŰSÍTÉS}

\textbf{\textit{\large félkövér kurzív nagy}}

$\backslash$ \{ \} \% \~{} \$ \_ \^{} \& \#
```



Bekezdések igazítása

```
\begin{flushleft}
flushleft
\end{flushleft}
\raggedright
raggedright

\begin{flushright}
flushright
\end{flushright}
\raggedleft
raggedleft

\begin{center}
center
\end{center}
\centering
centering
```



Listák

```
\section*{Sorszám nélkül}
\begin{itemize}
\item Első elem,
\item Második elem.
\end{itemize}

\section*{Sorszámmal}
\begin{enumerate}
\item Első elem,
\item Második elem.
\end{enumerate}

\section*{Definíciólista}
\begin{description}
\item[infámia] becstelenség, ...
\item[infámis] gyalázatos, ...
\end{description}
```

Sorszám nélkül

- Első elem,
- Második elem.

Sorszámmal

1. Első elem,
2. Második elem.

Definíciólista

infámia becstelenség, ...

infámis gyalázatos, ...

Képek és ábrák

```
\usepackage{graphicx}

\begin{figure}[h]
\centering
\includegraphics[scale=0.75]{latex.jpg}
\caption{LaTeX Project Logo}
\label{fig:latex}
\end{figure}
```

1. ábra. LaTeX Project Logo

- Képaláírás: \caption{}
- Lapon való elhelyezés opciói: [h] - pontosan itt, [t] - lap tetején, [b] - lap alján, [p] - külön oldalon Hivatkozási referencia: \label{}
- pl.: Ahogy az \ref{fig:ascii}. ábrán látható...

10. óra
LaTeX II.

Táblázatok #1

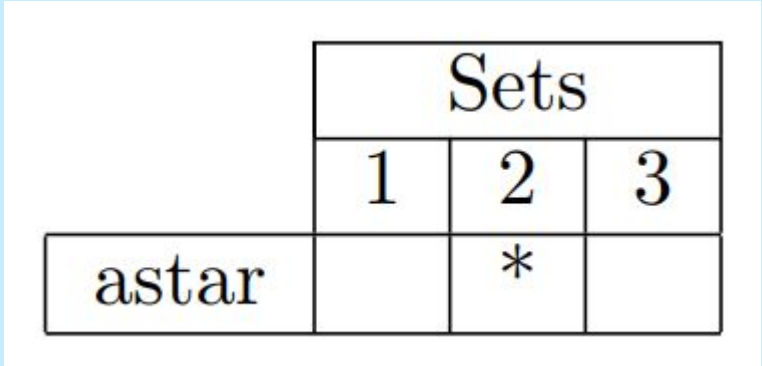
```
\begin{tabular}{|l|l|r|}
\hline
nap & min. & max. \\ \hline\hline
szerda & 9 & 16  \\ \hline
csütörtök & 6 & 15 \\ \hline
péntek & 8 & 14 \\  \hline
szombat & 6 & 8 \\ \hline
vasárnap & 3 & 6 \\ \hline
\end{tabular}
```

nap	min.	max.
szerda	9	16
csütörtök	6	15
péntek	8	14
szombat	6	8
vasárnap	3	6

- 1. sorban adjuk meg az igazítást (l, r, c)
- & : oszlop elválasztó karakter
- Az újsort és a vízszintes vonalakat is ki kell írni (\hline)

Táblázatok #2

```
\begin{tabular}{|c|c|c|c|}
\cline{2-4}
& \multicolumn{3}{|c|}{Sets} \\
\cline{2-4}
& 1 & 2 & 3 \\
\hline
\multicolumn{1}{|c|}{astar} & & * &
\\
\hline
\end{tabular}
```



	Sets		
	1	2	3
astar		*	

- \cline{n-m}: vízszintes vonal az n és m oszlopok között
- \multicolumn: oszlopok egyesítése

Táblázatok #3

```
\begin{table}[h]
\begin{tabular}{|p{5.5cm}|l|r|}
\hline
nap & min. & max. \\ \hline\hline
szerda & 9 & 16  \\ \hline
csütörtök & 6 & 15 \\ \hline
\end{tabular}
\caption{Táblázat címe}
\end{table}
```

nap	min.	max.
szerda	9	16
csütörtök	6	15

Table 1: Táblázat címe

- `p{5.5}`: rögzített oszlopszélesség
- `\caption{táblázat címe}`: táblázat alá írt szöveg

Képletek #1

```
\documentclass{article}
\usepackage[utf8]{inputenc
}
\usepackage{mathtools}
\begin{document}
$ x + 3 $

\begin{math}
y - 3
\end{math}

\begin{equation}
E=mc^2
\end{equation}

\end{document}
```

$x + 3$
$y - 3$

$$E = mc^2 \tag{1}$$

- \usepackage{mathtools} szükséges a képletekhez
- $ … $ között megadhatunk képletet
- \begin{math} és \end{math} között is megadhatunk képleteket
- \begin{equation} és \end{equation} között megadott képletek sorszámozva lesznek

Képletek #2

```
$y=c_2 x^2 + c_1 x + c_0$

$F_n = F_n-1 + F_n-2$

$F_n = F_{n-1} + F_{n-2}$

$\Omega = \sum_{k=1}^{n} \omega_k$

\begin{equation}
\Omega = \sum_{k=1}^{n} \omega_k
\end{equation}
```

$y = c_2x^2 + c_1x + c_0$
$F_n = F_n - 1 + F_n - 2$
$F_n = F_{n-1} + F_{n-2}$
$\Omega = \sum_{k=1}^{n} \omega_k$

$$\Omega = \sum_{k=1}^{n} \omega_k \tag{1}$$

- Alsó index: _, felső index: ^, csoportosítás: { }
- Görög betűk paranccsal hívhatóak elő (\Omega, \sum, …)

Képletek #3

```
\[ f(n) =
  \begin{cases}
    n/2       & \quad     \text{ha } n \text{ páros}\\
    -(n+1)/2  & \quad     \text{ha } n \text{ páratlan}\\
  \end{cases}
\]

\begin{center}
$ \frac{1}{2} $
\end{center}

\begin{equation}
x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
\end{equation}
```

$$f(n) = \begin{cases} n/2 & \text{ha } n \text{ páros} \\ -(n+1)/2 & \text{ha } n \text{ páratlan} \end{cases}$$

$$\frac{1}{2}$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{1}$$

- Elágazások: \[és \] között írjuk \-el kezdődnek a sorok is
- Törtek: \frac{számláló}{nevező}

Bibliográfia #1

cite.bib

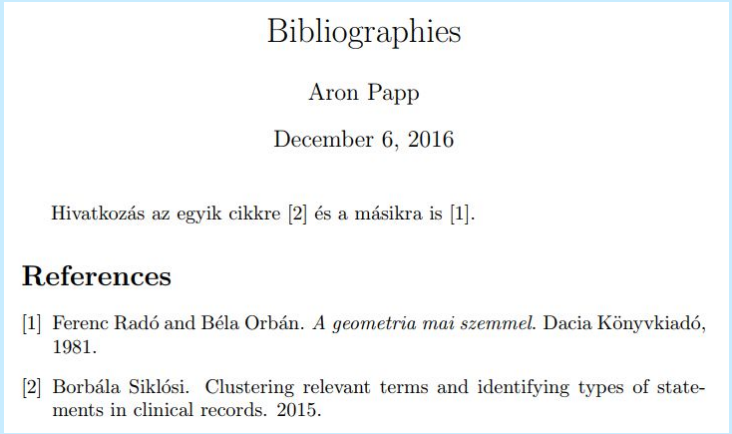
```
@book{radoorban,
author={Ferenc Radó and Béla Orbán},
title={A geometria mai szemmel},
publisher={Dacia Könyvkiadó},
year= 1981
}

@incollection{siklosi2015,
author={Siklósi, Borbála},
title={Clustering Relevant Terms and
Identifying Types of Statements in
Clinical Records},
year={2015}
}
```

- Léthehozunk egy .bib kiterjesztésű fájlt
- Itt készítjük el az adatbázist
- Számos mezőt használhatunk
  - author, title, publusher, year, isbn, booktitle, volume, series, editor, doi, publisher, keywords, pages, language, ...

Bibliográfia #2

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\title{Bibliographies}
\author{Aron Papp}
\date{\today}

\begin{document}
\maketitle

Hivatkozás az egyik cikkre
\cite{siklosi2015} és a másikra
is \cite{radoorban}.

\bibliographystyle{plain}
\bibliography{cite.bib}

\end{document}
```

- `\bibliographystyle{plain}` `\bibliography{cite.bib}`:
  - A .tex fájlban hibatkozunk a .bib fájlra és megadjuk a stílust is
- `\cite{azonosító}`
  - Beillesztjük az adatokat az adatbázisból



Bibliographies

Aron Papp

December 6, 2016

Hivatkozás az egyik cikkre [2] és a másikra is [1].

References

[1] Ferenc Radó and Béla Orbán. *A geometria mai szemmel.* Dacia Könyvkiadó, 1981.

[2] Borbála Siklósi. Clustering relevant terms and identifying types of statements in clinical records. 2015.

Előformázott szöveg

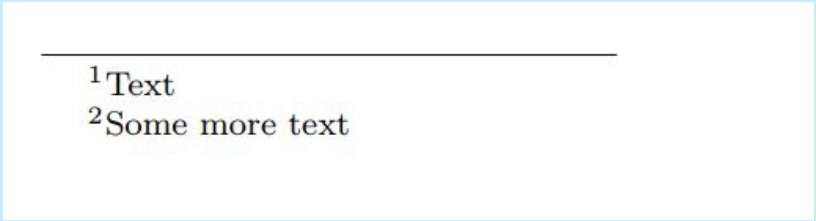
```
\usepackage{listings}

...

\begin{verbatim}
for i in range(0,10):
    if i < 3:
        print i
\end{verbatim}

\begin{lstlisting}[language=pytho
n]
for i in range(0,10):
    if i < 3:
        print i
\end{lstlisting}
```

- `\begin{verbatim} … \end{verbatim}` : normál előformázott szöveg
- `\begin{lstlisting}[language=python] \end{lstlisting}` : szintaxis kiemelés

```
for i in range(0,10):
    if i < 3:
        print i


for i in range(0,10):
    if i < 3:
        print i
```

Lábjegyzet

```
\footnote{Text}
\footnote{Some more text}
```



- A lábjegyzet alul egy horizontális vonal alatt jelenik meg sorszámozva

Szintaxisfák

```
\documentclass{article}
\usepackage[utf8]{inputenc}

\usepackage{tikz}
\usepackage{tikz-qtree}

\begin{document}

\begin{center}
\Tree [.S [.NP LaTeX ] [.VP [.V
is ] [.NP fun ] ] ]
\end{center}

\end{document}
```

- tikz és tikz-qtree csomagokat használjuk hozzá
- Csomópontot a pont jelzi
- Leveleknél nincs külön jelölés
- Szögletes zárójelekkel csoportosítunk