**Chapter 10**

File System Interface/ Fájlrendszerek – 12. előadás

File koncepciója/fogalma

* Összefüggő/folytonos logikai címtér
* Háttértár oldja meg a többit 🡪 Háttértárnak nem feltétlenül folyamatos, usernek igen
  + Itt jöhet be, mint probléma a töredezettség
* UNIX-os világban szó szerint minden fájl
  + Eszközök
  + Parancsok
  + Standard Kimenet/bement/error kimenet
  + Anyád
* File típusok 🡪 Lényegében végtelen van
  + Bináris
  + Szöveges
  + Végrehajtható
  + Ezek csak felhasználó számára fontosak a gépnek minden csak egy bytesorozat
  + Kiterjesztés 🡪 Plusz infó, nem kötelező (Windowson kinda)

Fájl tulajdonságai

* Név
* File ID 🡪 Külön azonosító a fájlrendszeren belül 🡪 Ez nem azt jelenti hol van a disken
* Típus 🡪 Néhány rendszernek kell
* Location 🡪 Hol van a fájl magán a disken
* Méret 🡪 byteban
* Védelmi lehetőség/Jogosultság
* Időbélyeg
  + Létrehozás ideje
  + Utolsó módosítás ideje
  + Utolsó hozzáférés ideje
  + Lehet több is, ez alap

File műveletek

* Létrehozás – Törlés
* Írás – Olvasás
* Pozícionálás fájlon belül 🡪 Seek 🡪 Létezik aktuális pozícióra mutató adat
* Truncate/Csonkolás 🡪 Kitörli a tartalmát, név, tulajdonság megmarad
* Open – Close

Megnyitott fájlok kezeléséhez szükséges adatok

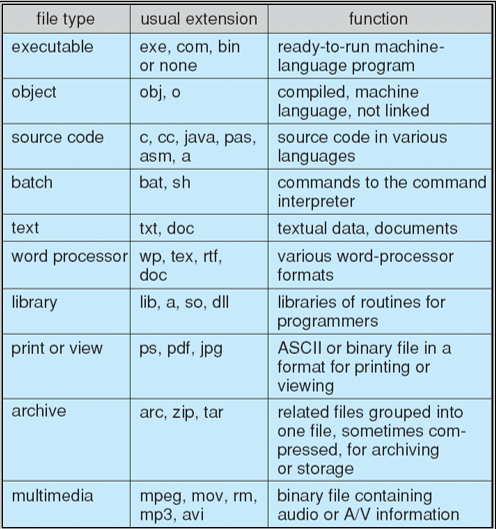
* Oprendszer nyilvántartja mi van megnyitva jelenleg
  + Ezért nem lehet átnevezni egy fájlt, ami meg van nyitva
* File Pointer 🡪 Hol vagyunk az aktuális fájlban
* Open számláló 🡪 Hányszor lett megnyitva
* Location
* Hozzáférési jog
* Amíg nincs konkurens hozzáférés addig ez pacek

Nyitott fájl Zárolás

* Olvasási Zár/Osztott zár 🡪 Akármennyi folyamat hozzáférhet egy fájlhoz
* Írási zár/Exkluzív zár 🡪 Én írok a többieknek kuss
  + Adatbázisok

Java dolgok nem kellenek innen

File kiterjesztések, típusok

* Executable
  + Linuxban nem kell
  + Windowsban igen
* Object 🡪 Tárgykód
* Batch
  + Minden rendszerben van
  + Linux.sh
  + Windows.bat
* Könyvtárak=Függvénygyűjtemények
  + Lib, dll

Fájl Struktúra

* Eddig 🡪 Összefüggő logikai címtér
* Hogyan olvassuk? 🡪 Byte-osával 🡪 Not good
  + Itt is mint fájlrendszerben ráépítünk egy logikai blokkot
    - Átvitel egység
  + Szöveges fájl 🡪 Átviteli egység ugyanaz, mint a bytesorozat
    - Bináris, egy kódtáblával
      * Ascii
      * UTX-8 stb.
  + Bináris fájl
    - Telefonkönyvet binárisan 🡪 Szövegesen / Struct-al C-ben
      * Struct   
        {  
        String name;  
        Int number;  
        }
    - Ha megmondom az átviteli egységet, akkor nem byte-onként olvas, hanem structonként 🡪 Egyszerre egy név-telefonszám párt olvas 🡪 Hatékonyabb, mint byte-onként
  + Az eddigi szekvenciális olvasást, hozzáférést felváltja a direkt hozzáférés
    - Logikai rekordok, 1 Struct
    - Speciális szekvenciális, ha úgy nézzük

Hogy kerül ki a diskre a fájl? - Directory Structure

* A Disk-et partíciókra tudjuk bontani
  + Ezeket a partíciókat köteteknek is nevezzük
  + Lemez egy része
* Cél a fájlok tárolása 🡪 Ehhez kell egy fájlrendszer
  + Fájlok rendszeres tárolására alkalmas megoldás
  + Vannak általános, és speciális célokra kifejlesztett fájlrendszerek

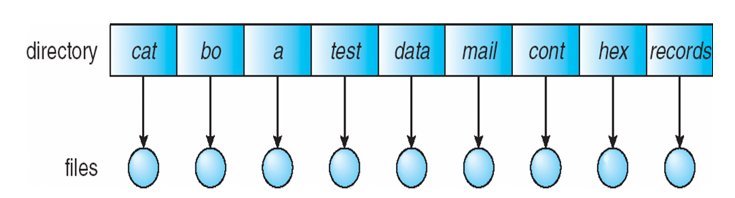
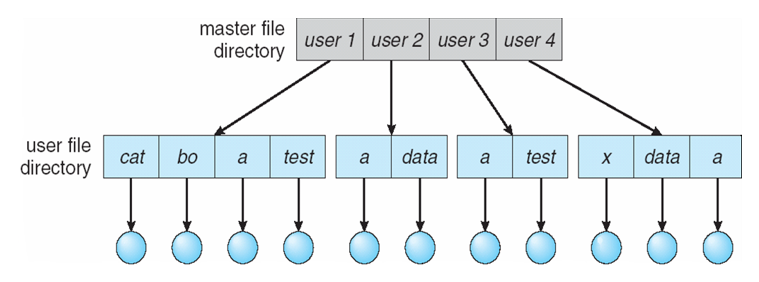
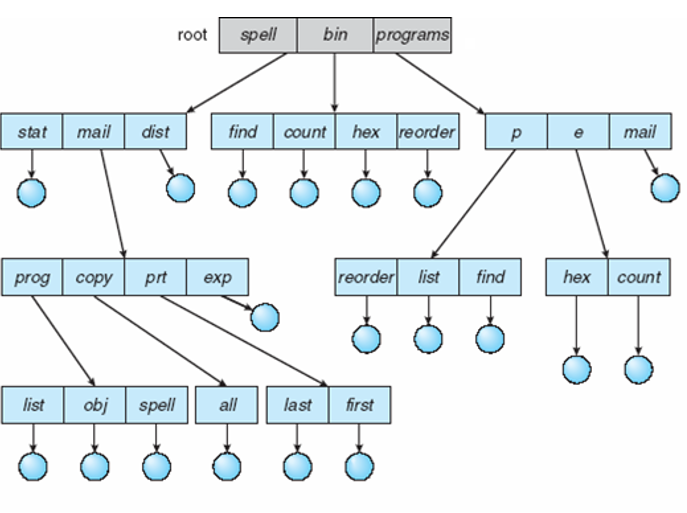
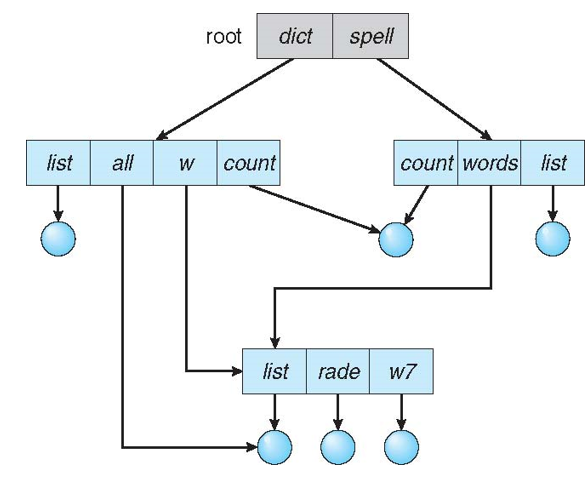
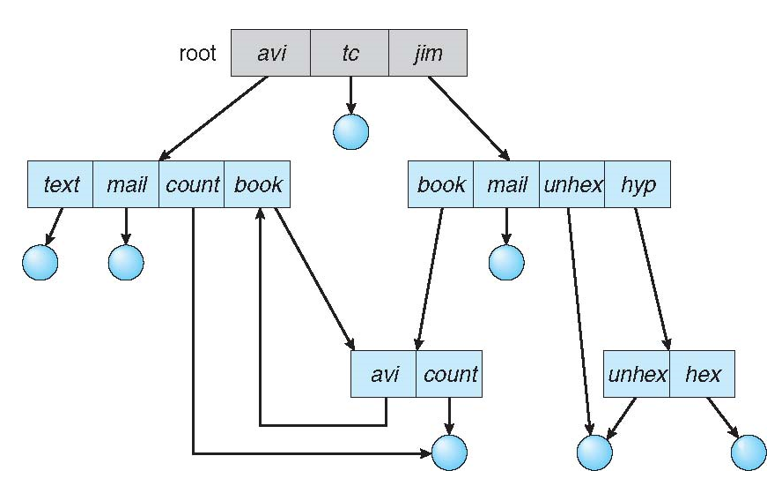
Fájlrendszertípusok

* Szükségszerűen, sok a különbség köztük
* Procfs 🡪 Minden fájl, folyamatok is
* Hálózati fájlrendszerek
  + Szükséges biztosítani a kötet távoli elérését
* Még sok más

Könyvtárműveletek

* File keresése
* Fájl létrehozása – Törlése
* Fileok listázása
* Fileok átnevezése
* Filerendszer átjárása

Directory Organization

* Kezdetben
  + 1 Partíció 🡪 1 Könyvtár
* Egyszintű
  + Kezdetben minden fájl egy mappában, az egész disk egy mappa
  + Nem lehet két ugyanolyan nevű file-om
  + Két felhasználó nem adhatja ugyanazt a filenevet
  + Amíg kevés file, addig jó, hatékony, gyors
  + Nincs lehetőség csoportosításra
* Kétszintű
  + Userenként kétszintű 🡪 Minden usernek van egy egyszintű kötete lényegében
  + Megjelenik az elérési útvonal
  + Lehet két ugyanolyan nevű fájl két külön usernek
  + Hatékony, gyors
  + Nincs lehetőség csoportosításra
* Fastruktúra
  + Egy gyökérmappa, és annak almappái
    - Minden mappának lehet almappája
  + Relatív és abszolút elérési útvonal megjelenése
  + Csoportosítás lehetséges
  + Gyors, hatékony keresés
  + A bejárása egy valós probléma lett
  + Hogyan oldjuk meg, hogy különböző elérési útvonalakon elérjünk ugyanoda?
* Aciklikus-gráf struktúra
  + Tud mindent, amit a Fa, csak megoldja a bejárást
  + Osztott listák, és alkönyvtárak
  + Ha valami két irányból közelíthető meg, hogy letöröljük mindkét irányból le kell törölni
    - Dangling Pointer problénája
      * Linkek megjelenése
* Általános Gráf Struktúra

Linkek

* Hard Link
  + Nem jön létre új fájl effektíve
  + Csak köteten belül működik
  + Egy pointer lényegében
* Soft link/szimbolikus link
  + Egy fájl amiben van egy elérési út
  + Lehet broken link, ha az adott helyen ahova mutat nem létezik az a fájl/mappa
  + Fájlrendszereken átívelhet 🡪 Nincs kötethez kötve

Filerendszer Csatolás

* UNIX alapú OS-nél csak becsatoljuk egy mappa alá és világbéke
  + Egy nagy fa van amihez hozzácsatoljuk a kötetet
* Windows
  + xd
  + Max 24 kötet lehet mert annyi maradt az ABC-ből
  + Lol

File Sharing/Megosztás

* Hogyan oldom meg a hozzáférést az állományaimhoz?
  + Jogosultságkezelés, csoportok
    - Linuxon pacek
    - Windows már megint nem tudom mi a faszt csinál
* Lokálisnak kell lennie?
  + Szokásos válasz🡪 Nem

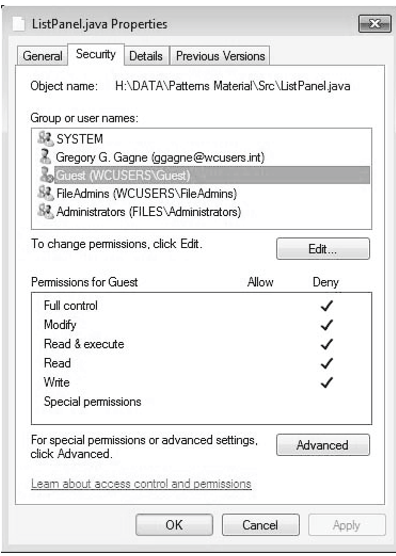
Remote File Systems

* Távoli kötetet csatolunk fel
  + Nincs lényegi változás
  + UNIX NFS
  + Windows CIFS

Védelem

* Hozzáférés típusai
  + Írás – Olvasás
  + KIVÉGZÉS
  + Hozzáfűzés
  + Törlés
  + Listázás

Hozzáférési listák és csoportok

* Linux
  + Owner
  + Group
  + Other/Guest/Public
* Windows
  + Whatever the fuck this is
* A Windowsos jó, de a Linuxos működik