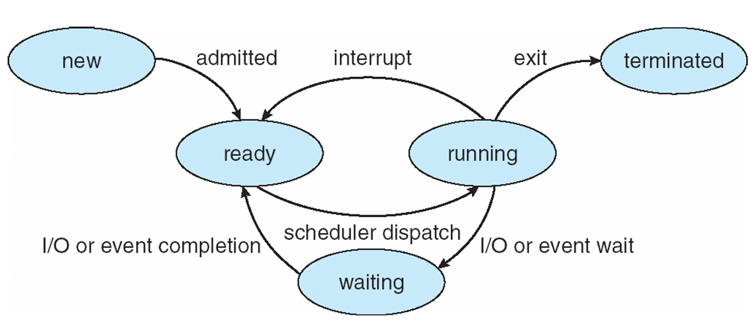
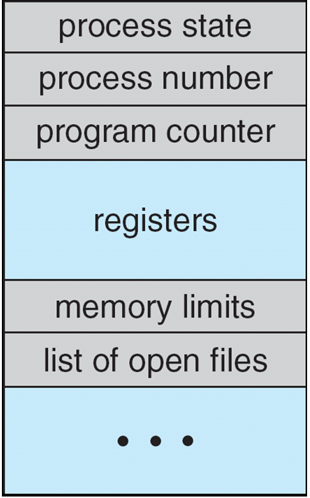
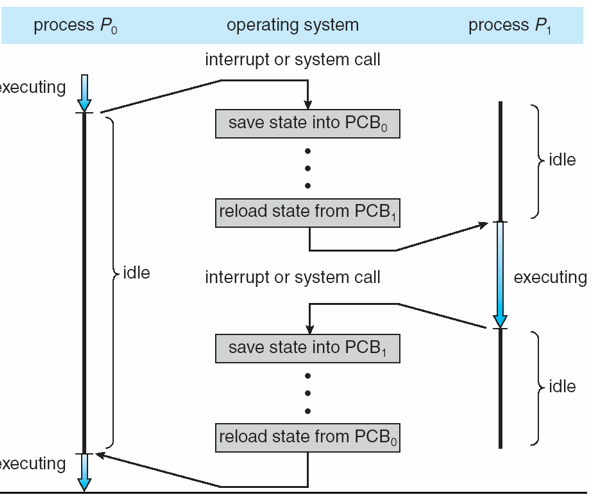
**Chapter 3 – 5. előadás**

Folyamatok

Folyamatok koncepciója

* Batch systems – jobs
* Jobs=Process=Folyamat
* Folyamat
  + Egy KIVÉGZÉS-ben lévő program
  + A KIVÉGZÉS szekvenciálisan kell történjen
  + A folyamat egy aktív entitású program
  + Egy program lehet egyszerre több folyamat is
* Folyamat részei
  + Utasítások, kódszegmens – Text
  + Utasításszámláló
  + 3 féle memóriaterület
    - Adatszegmens 🡪 Van mérete
      * Globális változók
    - Stack/verem
      * Lokális változók, függvényparaméterek
      * Hívási lánc
    - Heap/halom
      * Dinamikusan allokál memóriát a programhoz futási idő alatt
  + Regiszterek
  + Fordítók nem használják ki az oprendszer által nyújtott lehetőségeket, max 1 Mb az alap allokált memória 🡪 stack 🡪 Legtöbbször elég
    - 32 bit 🡪 Max 4 Gb memória
    - 64 bit 🡪 Max 16 Gb memória 🡪 Heap/Halom
    - Unix, Linux alatt Max Stack méret 🡪 8 Mb 🡪 Kernellel lehet változtatni
* Folyamat állapotai
  + Létrejövés/New
  + Futó/Running
  + Várakozó/Waiting
  + Készen/Ready
  + Vége/Terminated

Process Control Block – PCB

* Folyamat állapota
* Utasítászámláló
* Folyamatazonosító 🡪 Process Number
* CPU regiszterek
* Folyamatütemező
* Memóriakezelő/limit, allokált memória
* Nyomkövetési dolgok, Időlimit a program végrehajtására
* I/O állapot, allokált I/O eszközök
* A kernelnek ez a blokk a/egy folyamat
* Másodpercenként 8-9 folyamatváltás
* Multiprogramozottság ára 🡪 Idő
  + Folyamat állapotának elmentése 🡪 Másik folyamat betöltése 🡪 Megszakítás   
    Másik folyamat végrehajtása 🡪 Elmentése 🡪 eredeti folyamat visszatöltése

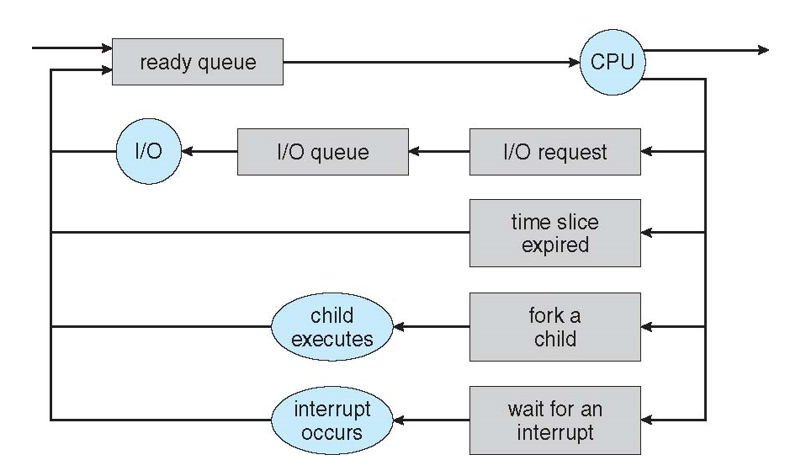
Szálak – Threads

* Eddig a PCB csak szekvenciális végrehajtásra volt képes 🡪 Mert 1 utasításszámláló
  + Most több utasításszámláló 🡪 Több „szálon futhatnak” folyamatok

Folyamathierarchia

* Gyökér-Szülő-Gyermek 🡪 Fa alakú 🡪 Gyökér +levelek
* Ősfolyamat 🡪 Elindítja az összes többi folyamatot
* 1 folyamatnak 1 szülőfolyamat, bármennyi gyerekfolyamat
* Fork 🡪 Folyamat elindít egy másik folyamatot
  + Megvárja a szülő folyamat a gyermek folyamat lefutását
* Ha a szülő folyamat leáll 🡪 Ki kell gyalulni az összes alatta lévő gyermek folyamatot
  + Orphan/árva folyamatok 🡪 Keresni kell új szülőt 🡪Gyökérfolyamat 🡪 init 🡪 Időnként elindít egy wait rendszerhívást, az árva folyamatok lefutnak, vagy ha nem kigyalulja a megmaradt folyamatokat a kernel
  + Egész fának valahogy el kell tűnnie
* Ha a szülő nem foglalkozik azzal hogy a gyerek „kész van”, azaz nem hívja meg a wait-et, a folyamatból Zombie folyamat lesz
  + Leadja az erőforrást, terminált a folyamat, de megmarada a PCB blokkja

Folyamatütemezés

* Cél 🡪 Maximalizálni a gépidőt, folyamatok közötti váltás ennek elérése érdekében
* Ütemező 🡪 Választ melyik folyamatot végezze el következőleg
* Sorban vannak a programok 🡪 Versengenek az erőforrásért
  + Ütemező láncolt listából választ
* Kernel listákat kezel, folyamatokból fát épít
* Ütemező sorok
  + Job queue
    - Az összes folyamat a rendszerben
  + Ready queue
    - Az összes folyamat, ami készen áll a KIVÉGZÉSRE
  + Device queue
    - A folyamatok, amik I/O eszközre várnak
  + A folyamatok ezek között lépkednek
* Context switch
  + „A” folyamatról váltunk B folyamatra
  + A fut 🡪 A ment 🡪 B betöltődik 🡪 B fut 🡪 B ment 🡪A betöltődik 🡪 A fut

Ütemezők

* Rövidtávú
  + Laptop, telefon, PC 🡪 Round Robin
  + Kiválasztja melyik programot futassa
  + Néha az egyetlen ütemező a rendszerben
  + Ő a sebesség
* Hosszútávó ütemező
  + Lassú 🡪 Akár percek
  + Kiválasztja melyik program kerüljön ready queue-ba
  + PCB sokkal nagyobb ablak
  + Folyamatosan becsülgetnek kinek lenne célszerű adni gépidőt
    - Folyamatok kategorizálása
      * I/O vagy CPU intenzív
      * Hogy maximalizálja mindkettőt
  + Folyamatos terhelés
  + Régi mainframek

5. Előadás vége

Böngésző példák

* Van egy pár böngésző, ami egy folyamatként működik
* Chrome
  + Minden lap egy külön folyamat, amit nyit

Folyamatok lehetnek

* Különálló, függetlenek
* Kooperáló
  + Hatással lehetnek rá más folyamatok, és hatással lehet más folyamatokra
  + Okok erre
    - Információ megosztás
    - Számolás felgyorsítása
    - Modularitás
    - Kényelem
  + InterProcess communication 🡪 IPC
    - Osztott memória
    - Message Passing
* Producer- Consumer Process
  + Producer 🡪 Előállít információt
  + Consumer 🡪 Megkapja az információt
    - Van egy buffer ami lehet
      * Limit nélküli
      * Limitált

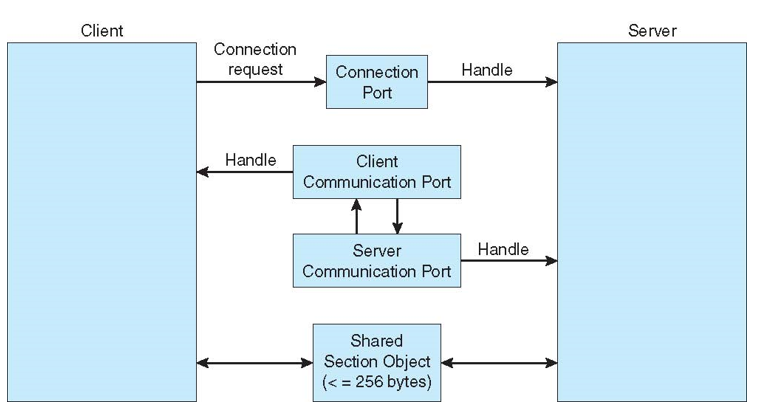
Interprocess Communication

* Osztott memória
  + Egy adott memória rész meg van osztva folyamatok között
  + Ez a kommunikáció felhasználói folyamatok irányítása alatt van, nem az OS irányítása alatt
  + Fő feladat 🡪 Egy mechanizmus arra, hogy felhasználói folyamatok tudjanak szinkronizálni egymás között amikor osztott memóriát használnak
* Message Passing 🡪 Üzenet átadás
  + Egy másik mód arra, hogy folyamatok egymás között tudjanak kommunikálni
  + Osztott változókkal teszik meg ezt a kommunikációt
  + Send – Receive 🡪 Communication link a folyamatok között
  + Message size 🡪 Fix, vagy változó
  + Communication link implementációk
    - Fizikai
      * Osztott memória
      * Hardver busz
      * Hálózat
    - Logikai
      * Közvetlen, vagy közvetett
      * Közvetett
        + Mailboxok használata 🡪 Portok
        + Minden mailboxnak saját ID
        + Csak akkor van link két folyamat között, ha egy mailboxhoz tartoznak
      * Szinkronizált, nem szinkornizált
      * Automatic or explicit buffering
  + Közvetlen link
    - Send, Receive
    - Linkek automatikusan jönnek létre
    - Egy linkhez pontosan 2 folyamat van, es 2 folyamathoz 1 link
  + Közvetett link
    - Mailboxok használata 🡪 Portok
      * Minden Mailboxnak saját ID
      * Csak akkor tud két folyamat kommunikálni, ha ugyanahhoz a mailboxhoz csatlakoznak
    - Tulajdonságai
      * Csak akkor jöhet létre, ha egy a mailbox-uk
      * Egy link több folyamathoz is tartozhat
      * Minden folyamatpárhoz akár több link is tartozhat
      * Lehet Egy vagy Kétirányú
    - Műveletek mailboxokkal
      * Új létrehozása
      * Üzenet fogadása, küldése
      * Mailbox törlése
      * Alapművelet🡪 send, receive
    - Probléma
      * P1 küldi üzenetet mailboxra, amihez P2 és P3 csatlakozik
        + Ki kapja meg az üzenetet?
      * Megoldás
        + Egy link max 2 folyamathoz tartozhat
        + Egyszerre egy folyamat hajthat végre receive operationt
        + Kiválasztunk önkényesen egy receivert

Sender megkapja ki kapta az üzenetet

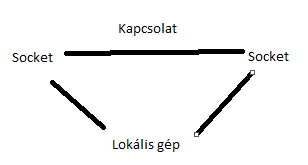
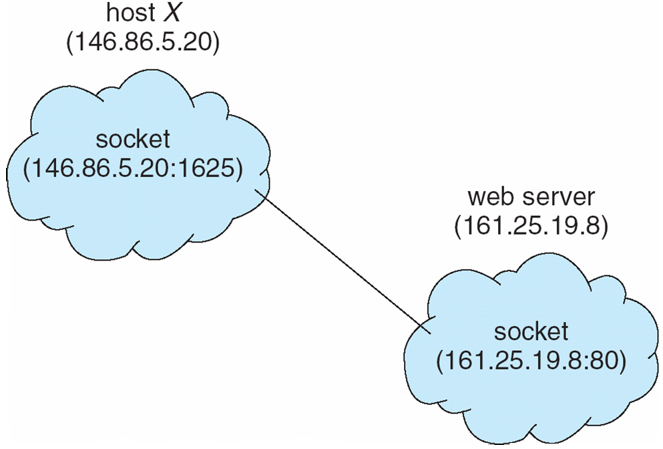
* Szinkronizáció
  + Message Passing lehet Blocking vagy Non-Blocking
  + Blocking 🡪 Szinkronizált
    - Blocking send 🡪 a küldő blokkolva van, amíg nem kapja meg a receiver
    - Blocking Receive 🡪 A receiver blokkolva van, amíg nem elérhető az üzenet
  + Non – Blocking 🡪 Aszinkronizált
    - Non-Blocking send 🡪 A küldő elküldi az üzenetet, és megy is tovább, nem vár
    - Non-Blocking receive
      * A receiver vagy érvényes üzenetet kap
      * Vagy null üzenetet
  + Ha send és receive is blokkolt akkor 🡪 Randevú
* Buffering
  + Üzenetet sora (queue)
  + Implementáció háromféle lehet
    - Zero capacity 🡪 Nincs egy üzenetsor 🡪 Küldőnek várnia kell a receiverre
    - Bounded capacity 🡪 Véges számú üzenet tárolása
      * Ha tele van a sender vár
    - Unbounded capacity 🡪 Végtelen számú üzenet tárolása
      * Sender soha nem vár

IPC rendszer példák

* POSIX osztott memória
  + Mach
* Windows

6. Előadás vége

Socket

* Unix 🡪 Miért legyen minden lokálisan?
  + Erre a kérdésre mindig az a válasz, hogy nem kell annak lennie
  + Host 🡪 Socket 🡪 Kapcsolat 🡪 Socket 🡪 Webserver
  + Lokálisan is lehet két socket között kapcsolat, azaz lehet kliens a saját gép is
  + Portok használata

Remote Procedure Calls - RPC

* Egy nyelv, amiben le lehet írni a végződéseket, socketeket
* Socket interface újragondolva
* Szükség van egy third party szoftverre 🡪 Az adott nyelven elérhetővé tegye a rendszerhívásokat
  + ORB – Object Request Broker
    - Segít megtalálni a távoli helyen a végpontot
* Socket interface egy absztrakciója
* Windowson – Matchmaker
* IPC 🡪 Gépen belül
* RPC 🡪 Gépen kívül

Csővezetékek

* Egy folyamat standard outputját összedrótozza egy folyamat standard inputjával
* Mivel minden fájl, ezek is
  + Három eszközfájl lényegében
* Fd 🡪 File descriptor
* Névtelen és nevesített csővezeték

7. előadás eleje