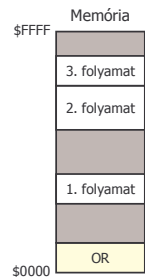


## Folyamatkezelés a modern operációs rendszerekben

Alapfogalmak  
Folyamatok életciklusa és állapotai  
Folyamatmodellek  
Ütemezés  
Megvalósítási példák (UNIX, Windows 2000)

## A folyamat fogalma

- A folyamat
  - Végrehajtás alatt álló, a virtuális memóriába betöltött, környezetével kölcsönhatásban lévő programpéldány
- Egy gép – egy vagy több folyamat
  - Erőforrások jobb kihasználása
  - Kényelem

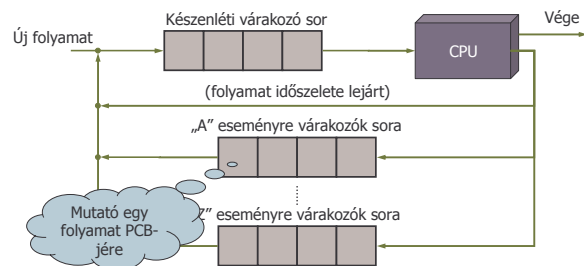


## Folyamatok nyilvántartása

- Folyamatleíró blokk (PCB)
  - Azonosító (ID)
  - Állapotjelző
  - Regiszterek tartalma (programszámláló is)
  - Memória- és erőforrásadatok
  - Prioritási és elszámolási információk
  - Egyéb mutatók
- Folyamatábrázoló (PT)
  - Logikailag tartalmaz minden PCB-t
    - Táblázat vagy lista belső mutatókkal

## Folyamatok nyilvántartása

- Várakozó sorok
  - Minden állapothoz egy vagy több sor

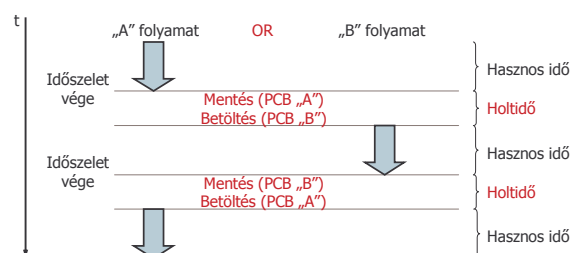


## Folyamatok életciklusa (1)

- Létrehozás
  - Betöltés
    - Betöltés háttértárról
    - Betöltés memóriából
  - Címter hozzárendelése
    - A „gyermek” (új) folyamat új programot tartalmaz
    - A „gyermek” lemásolja a „szülő” (létrehozó) folyamat címtérét
  - Végrehajtás elkezdése
    - A „szülő” folyamattal párhuzamosan
    - A „szülő” megvárja a „gyermek” befejeződését

## Folyamatok életciklusa (2)

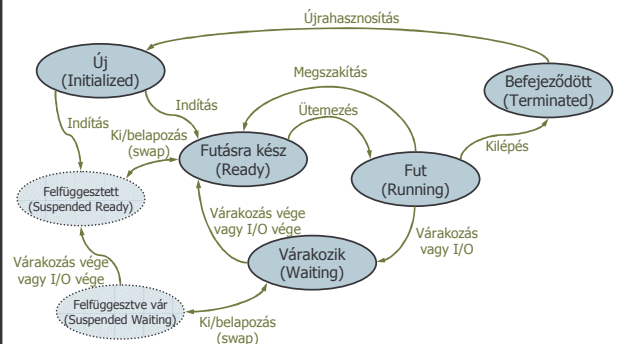
- Kontextusváltások
  - Az OR felcseréli a futó folyamat kontextusát egy másikéval (mentés és betöltés)



## Folyamatok életciklusa (3)

- Befejeződés
  - A folyamat befejezte feladatát
  - A folyamat „szülője” befejeződött és az OR ilyen helyzetben nem engedi tovább létezni a „gyermekeket”
  - A folyamat erőforráskorlátba ütközött
  - A folyamat meg nem engedett műveletet hajtott végre

## Folyamatok állapotátmenetei



## Folyamatmodellek (1)

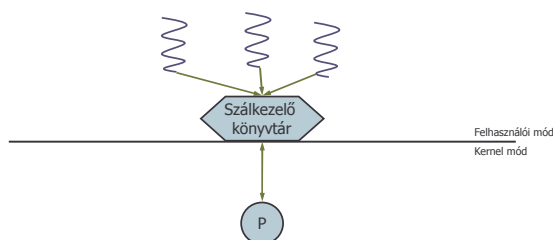
- Folyamat alapú modell
  - Csak folyamatok léteznek benne
  - Minden feladat megoldható ezzel a modellel
    - Ütemezés, párhuzamos futtatás
    - Kommunikáció (IPC, memória megosztása)
    - Aszinkron I/O műveletek
  - Teljes kontextusváltások
  - Kiváló belső védelem
    - Külön címtér, külön kontextus

## Folyamatmodellek (2)

- Folyamat-szál alapú modell
  - Szál: „pehelysúlyú folyamat”
    - Programfuttatás egysége
    - Saját programszámláló, regiszterek, verem
  - A szálak a folyamatokon belül léteznek
    - Osztóznak a folyamat címtérén
    - Kvázi- és valódi párhuzamosság is megvalósítható
  - Előnyei:
    - Jelentős teljesítménynövekedés
      - Csak jól megírt programok esetén
    - Kényelmesen kezelhető

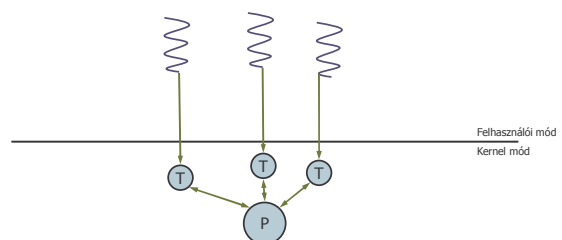
## Szál alapú modell megvalósítása

- Felhasználói módú szálkezelés
  - Megvalósítás függvénykönyvtárként
  - Az operációs rendszer nem foglalkozik a szálakkal



## Szál alapú modell megvalósítása

- Kernel módú szálkezelés
  - Hasonló API-k folyamatokhoz és szálakhoz
  - Szálkezelés az operációs rendszer szintjén



## Ütemezés

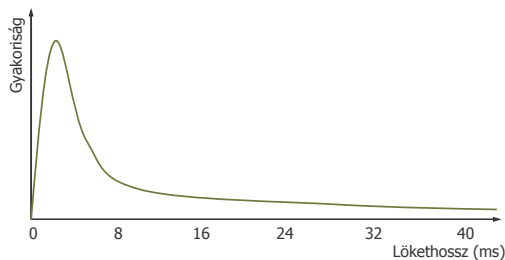
- Rövid távú
  - Folyamatválasztás minden kontextusváltásnál
  - Rendkívül gyors algoritmusra van szükség
    - A gyakorlatban kombinált algoritmusokat használnak
- Középtávú
  - Memóriagazdálkodási szempont
- Hosszú távú
  - Folyamatok beengedése a rendszerbe az aktuális állapot függvényében
    - Felhasználónkénti maximális folyamatszám
    - Rendszerszintű maximális folyamatszám

## Rövid távú ütemezés

- Célfüggvények:
  - Legjobb processzorkihasználás
  - Legnagyobb teljesítmény
    - Időegységenként befejezett folyamatok száma
  - Legkisebb teljes végrehajtási idő
    - Átlagos lassulás mértéke
  - Legrövidebb várakozási idő
    - Átlagos várakozási idő a készenléti állapot sorában
  - Legrövidebb válaszidő

## CPU löket

- A folyamat két, egymást követő várakozó állapota között eltelt idő



## Rövid távú ütemezés algoritmusai

- „First Come, First Served” (FCFS) algoritmus
  - Más néven FIFO algoritmus
  - Végrehajtás érkezési sorrendben
- „Shortest Job First” (SJF) algoritmus
  - Legrövidebb CPU löketre készülő folyamat végrehajtása elsőként
  - Típusok:
    - Nem preemptív
      - Nem szakítható meg az éppen futó folyamat
    - Preemptív
      - Ha rövidebb CPU löketű folyamat érkezik, ő fut

## Rövid távú ütemezés algoritmusai

- Prioritásos ütemező algoritmus
  - Minden folyamat rendelkezik egy prioritással
  - Mindig a legnagyobb prioritású folyamat fut
  - Típusok:
    - Nem preemptív
      - Nem szakítható meg az éppen futó folyamat
    - Preemptív
      - Ha magasabb prioritású folyamat érkezik, ő fut
  - Súlyos probléma: kiéheztetés
    - Alacsony prioritású folyamatok nem jutnak szóhoz
    - Megoldás: pl. öregítés (régén futott folyamatok prioritásának fokozatos növelése)

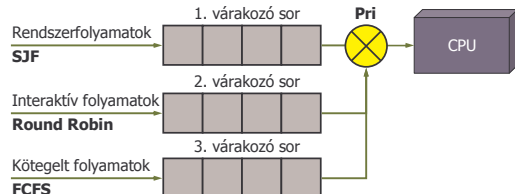
## Rövid távú ütemezés algoritmusai

- „Round Robin” (körbenforgó) algoritmus
  - Minden folyamat kap egy időszletet ( $q$ )
    - Általában 10–100 ms közötti időtartam
  - Az időszlet lejártával lekerül a processzorról és a készenléti sor végére kerül
  - $n$  folyamat esetén minden folyamat a processzoridő  $1/n$ -ed részét kapja
    - A maximális várakozási idő  $(n-1) \cdot q$
  - Teljesítmény:
    - Nagy időszlet  $\approx$  FCFS
    - Kis időszletnél nagyobb a holtidő



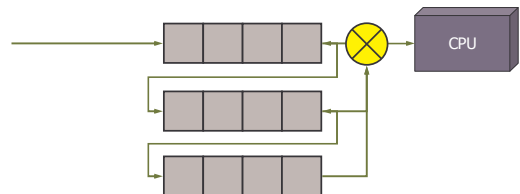
## Többszintű rövid távú ütemezés

- „Multilevel Queue” algoritmus
  - Több készenléti sor
    - Saját ütemezési algoritmus
  - Sorközi ütemezés (más algoritmussal)
    - Pl. fix prioritásos, soronkénti időszeleles



## Többszintű rövid távú ütemezés

- „Multilevel Feedback Queue” algoritmus
  - A folyamatok mozoghatnak a sorok között
  - Számos független paraméter
    - Sorok száma, sorok ütemezési politikája, folyamatok mozgatható feltételei, folyamatok kezdősora, sorok közötti ütemezés algoritmus...



## Példa – BSD Unix

- „Multilevel Feedback Queue” algoritmus
  - 32 különböző prioritású sor (0–7: rendszer-, 8–31: felhasználói folyamatok)
  - Minden folyamatnak van egy „nice” paramétere
    - Ez dönti el, melyik prioritási sorban várakozik
  - Soron belül körbenforgó ütemezés
  - Prioritás újraszámítása kb. 40 ms-onként (időseletenként átlagosan kb. 1-szer)
  - A felhasználói folyamatok ütemezése preemptív, a kernelfolyamatok ütemezése nem preemptív (!)

## Példa – Windows 2000

- „Multilevel Feedback Queue” algoritmus
  - 32 különböző prioritású sor
- Négy szintű folyamat-szál alapú modell
  - Szálkezelés az operációs rendszer szintjén
- Preemptív, prioritásos, körbenforgó ütemező
  - Szigorú, de dinamikus prioritásos ütemezés
  - A prioritások ideiglenes kiigazításával oldja meg a kiéheztetés problémáját
- További részletek a gyakorlaton