Előadás_#06.

1. Holtpont, Éheztetés

   A holtpont részletes tárgyalása előtt nagyon fontos leszögezni a következőt:

   - Az éheztetés folyamat szintű probléma.
     Egy folyamat akár a végtelenségig is várhat erőforrásra (pl. öregítés nélkül).
   - A holtpont operációs rendszer szintű probléma.
     Akár az egész rendszer megállhat, lefagyhat, (de ez nem minden esetben következik be, maradhatnak futó folyamatok) legrosszabb esetben azonban semmi sem tud futni. Amennyiben az összes erőforrás holtpontba kerül, a rendszer működése biztosan leáll.

   A hosszú távú ütemező indítja sorba a folyamatokat, melyek ezután az indulásukhoz szükséges erőforrásokra igényt jelentenek be. Azokat a folyamatokat, melyeknek csak a CPU – mint erőforrás hiányzik, az összes induláskor ismert erőforrás biztosítása után a rövid távú ütemező várakozó sorba helyezi, azaz ezek a folyamatok futásra kész állapotba kerülnek. Amint a folyamat megkapja a CPU-t is a folyamat futása elkezdődik. Erőforrásigény azonban futás közben is keletkezhet. Például ha egy folyamat sikeres befejezését egy adott dallam lejátszásával is kívánjuk jelezni, nyilván a hang megszólalásáért felelős erőforrást (hardveresen a hangkártyát, szoftveresen a hangkártya meghajtó programjait – a hangkeverés lehetőségétől most ebben a példában tekintsünk el) nem célszerű már a futás elején lefoglalni, hiszen a folyamatunk futása közben más folyamatok hasonló célból, különböző hangjelzésekkel tudathatják lefutásukat.
   A preemtivitás, mint tulajdonság az erőforrások szempontjából is az egyik csoportosítási szempont.

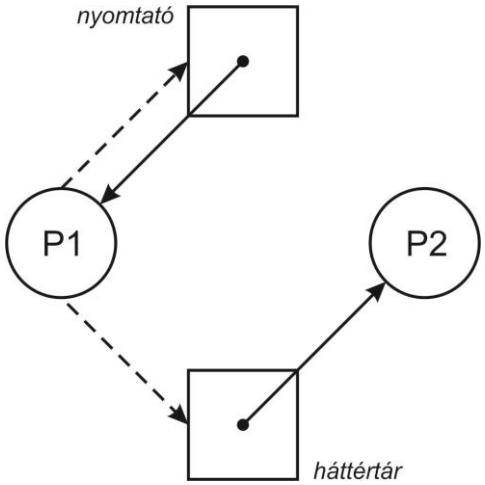
   - Megszakítható (Preemptív) erőforrás
     pl. processzoridő, memória
   - Nem megszakítható (nem preemtív) erőforrás
     pl. állományok, nyomtató, rajzgép, szalagos egység.

   A nem megszakítható hozzáférési mód olyankor szükséges (esetleg elengedhetetlen), amikor egy folyamat egy erőforráson olyan műveletet végez, amelyet nem szeretne, hogy egy másik folyamat megzavarjon. Normális esetben egy erőforrás csak akkor szabadul fel, ha a folyamat önként lemond róla, vagy a folyamat futása befejeződik.

A valóságban az erőforrások osztályokba sorolhatók, melyek az adott osztályon belül egyenértékűek, csereszabatosak. Az erőforrás osztályok, azaz egy másfajta csoportosítás:

- az egypéldányos osztályban csak egy erőforrás példány van
  (pl. nyomtató, plotter)
- a többpéldányos osztályban olyan erőforrás van, amit egyszerre több folyamat is használhat
  (pl. memória)

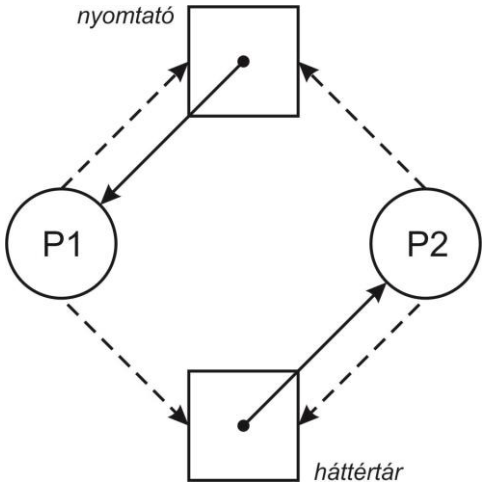
Probléma alapvetően a kizárólagosan használható, nem megszakítható erőforrásokkal van, vagyis a probléma (azaz az előre nem kalkulálható várakozás) akkor keletkezik, amikor a „P1” folyamatnak – már futása közben – szüksége van egy másik folyamat a „P2” által használt, kizárólagosan használható erőforrásra. Amennyiben „P1” prioritása kellően magas, magasabb, mint „P2” prioritása, akkor a kérdés az, hogy a „P2” folyamat állapottere menthető, vagy nem menthető. Ha nem menthető, és a folyamatot kilőjük (Kill) – hogy hozzáférjünk az általa lefoglalt kizárólagosan használható erőforráshoz – akkor az biztos, hogy adatvesztéssel jár, a „P2” adatai, eredményei elvesznek, így azokat a későbbiekben nyilván pótolni kell, amennyiben az egyáltalán lehetséges. Ha menthető, akkor a középtávú ütemező a „P2” folyamatot kitüntetett várakozó állapotba teszi, és „P1” végeztével „P2” folytatja futását.



A szaggatott vonalú nyíl, ami egy folyamattól indul és egy erőforrás határvonalára mutat, az úgynevezett "kérési él", azaz az erőforrás foglalásának szándékát jelzi. A folyamatos vonalú nyíl, ami egy erőforrás belsejéből indul és egy folyamat határvonalára mutat, az úgynevezett "foglalási él", azaz az erőforrás adott folyamat általi foglaltságát mutatja.

A fenti eset holtpontot nem okozott, de előre nem kalkulálható várakozást igen, sőt esetleg adatvesztést is (lásd „P2” menthetősége vagy újrakezdése).
[Szerver gépek esetében gyakorlatilag minden HW erőforrás többpéldányos és jellemzően cache-elhető is.]

Amennyiben a létező véges számú, jellemzően nem megszakítható erőforrás birtoklásáért egyszerre több folyamat is versenyben van, elő tud állni egy olyan állapot, amelyben lesz olyan „Px” folyamat, melynek az egyik általa kért erőforrás nem biztosítható, mivel azt már egy másik „Py” folyamat használja. Ez esetben a „Px” folyamat várakozni kényszerül. Szerencsés esetben relatíve rövid várakozás után hozzá tud férni az áhított erőforráshoz, amint az felszabadul. Szerencsétlen esetben megtörténhet az, hogy „Py” is várakozik, hiszen az általa óhajtott erőforrást egy harmadik folyamat, a „Pz” birtokolja, aki pedig a „Px” egy másik erőforrására vár – sőt akár sokkal hosszabb hasonló láncolat is kialakulhat.
A legszerencsétlenebb eset a körkörös várakozás, ami a holtpont egyik oka. A körkörös várakozás kialakulásához akár két folyamat is elegendő lehet.



A fenti esetben mindkét folyamat nyomtatni szeretne, amihez az adatokat a háttértárról olvasnák be. Jelen esetben a folyamatok az erőforrásokat nem azonos sorrendben igényelték. A „P1” folyamat először foglalást jelentett be a nyomtatóra, amit meg is kapott. A „P2” folyamat először foglalást jelentett be a háttértárra, amit meg is kapott. Ezután a „P1” hiába jelent be foglalást a háttértárra, azt már birtokolja a „P2”, és hasonló az eset a „P2” szempontjából is, hiszen a „P2” hiába jelent be foglalást a nyomtatóra, azt már birtokolja a „P1”. Azonos prioritás esetén nincs remény a továbblépésre.

Elvi szintű, azaz elvileg (főleg a ma jellemző memória konfiguráció mellett csak nehezen) elképzelhető az a példa is, amikor a holtpont azért alakul ki, mert egy folyamatnak valamilyen driver (azaz meghajtó program) használata szükséges egy adott erőforrás eléréséhez, de nincs a folyamathoz rendelt memóriában elegendő hely a meghajtó program betöltéséhez és végrehajtásához.

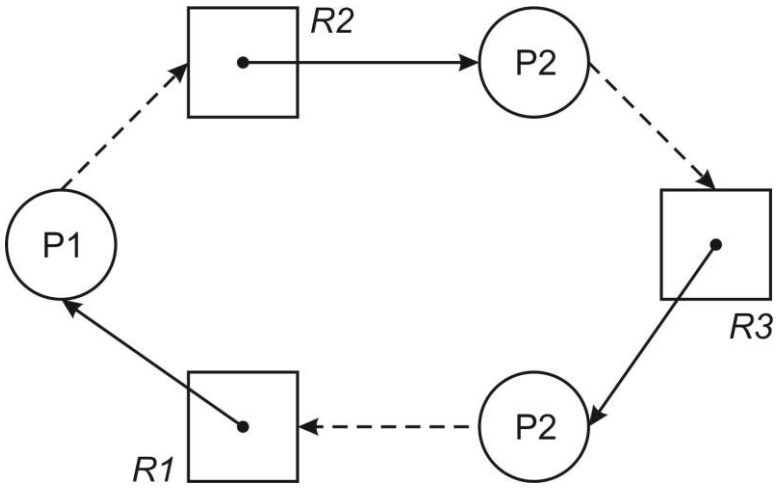
Az erőforrások használatba vételének és használatának lépései:

1. Igénylés.
   Amennyiben az igény nem teljesül, a folyamat várakozni kényszerül.
2. Az erőforrás kizárólagos használatának megkezdése.
3. Az erőforrás felszabadítása a folyamat végeztével.

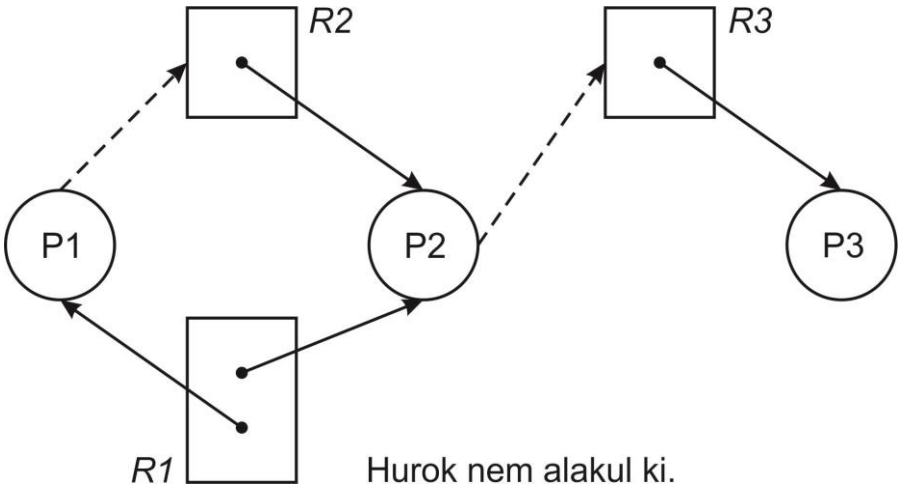
Ahhoz hogy a holtpont ki tudjon alakulni négy szükséges feltételnek kell egyszerre teljesülni:

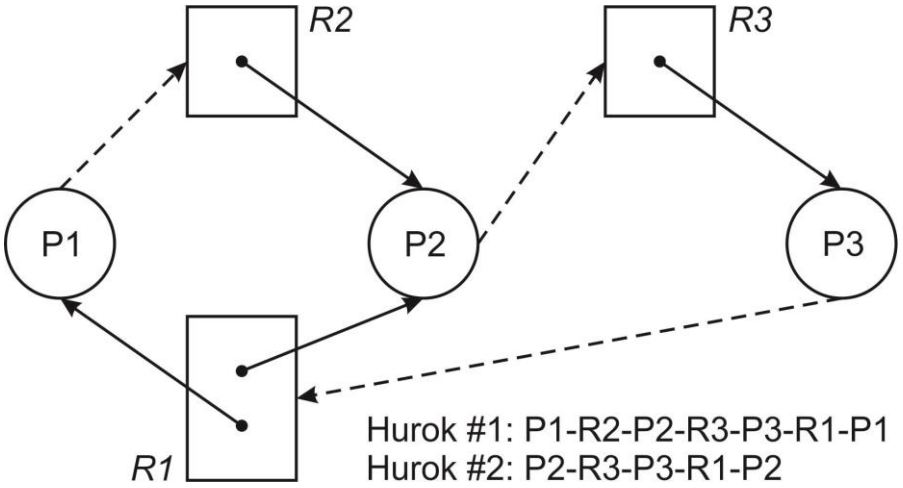
1. Kölcsönös kizárás
   Ezen feltétel szerint egy erőforrást egy időpillanatban csak egy folyamat használhat. Amennyiben egy folyamat igényt tart erre az erőforrásra, várakozni kényszerül.
2. Foglalás és várakozás
   Ez a feltétel azt jelenti, hogy legalább egy olyan folyamatnak kell léteznie, amely már rendelkezik egy erőforrás foglalással, de további olyan erőforrásokra vár, melyeket más folyamatok foglalásba tartanak, használnak.
3. Kiszorításmentesség
   Ez a feltétel azt jelenti, hogy a lefoglalt erőforrást az azt foglaló folyamattól másik folyamat elvenni nem tudja, azaz az erőforrás nem preemtív, az erőforrás csak a folyamat befejezése után szabadul fel.
4. Körkörös várakozás
   Ez a feltétel szerepelt a 2. oldalon („Px” „Py” „Pz” folyamatokkal modellezve).

Az erőforrás gráfok használatával tudjuk legkönnyebben átlátni a helyzetet. Általában az erőforrás-használati gráfban a hurok szükséges feltétele a holtpontnak. Amennyiben valamennyi erőforrás osztály csak egyetlen erőforrást tartalmaz, a hurok elégséges feltétel is egyben.



Azok az erőforrás osztályok, melyeknek az erőforrásaihoz egy időben több folyamat is csatlakozhat, jelentősen csökkentik a hurok kialakulásának valószínűségét. Amennyiben az erőforrás használati gráf nem tartalmaz hurkot, akkor biztosan nincs holtponti helyzetben.





A bármely okból kialakult holtpont feloldásához, megszüntetéséhez mindenképpen arra van szükség, hogy a holtpontot okozó erőforrás foglaltsága valahogy megszűnjön.

Az adott probléma (holtpont elkerülése) megoldásakor tekintettel kell arra lenni, hogy a probléma milyen áron (mennyi közvetlen adatvesztéssel, adott folyamat vagy folyamatok teljes újraindításával) oldható meg. A holtpont jellemzően nem számolható fel veszteség nélkül.

A holtpontot egyszerűbb megfelelő adminisztrációval megelőzni, mint a már kialakult holtpontot kezelni, megszüntetni. Nézzük meg, hogy a fent ismertetett négy szükséges feltétel közül melyik az, amelyik kiküszöbölhető:

1. Kölcsönös kizárás

   Nem megosztható erőforrások esetében nem küszöbölhető ki.

   Megosztható erőforrások viszont nem is követelik meg a kölcsönös kizárást.

2. Foglalás és várakozás

   Ez kizárható, hiszen azt jeleni, hogy egy folyamat nem kérhet új erőforrást, amíg lefoglalva tart egy másikat. A megvalósításnak két módja van:

   - Futás elején lefoglalja az összes erőforrást, és csak akkor futhat, ha valamennyi erőforrás a rendelkezésére is áll.
   - Az erőforrás-foglalás előtt a foglalt erőforrásokat felszabadítja, vagyis a folyamat csak akkor igényelhet egy erőforrást, ha már más erőforrást nem használ.

   A módszer biztos, de nem hatékony, hátránya, hogy rossz kihasználtságot eredményez, és nagy a kiéheztetés esélye is.

3. Kiszorításmentesség

   A nem preemtív erőforrás közvetlenül nem vehető el a folyamattól, de ha a folyamat további erőforrásra vár, azaz várakozik, futni képtelen, akkor célszerű az egész folyamatot megállítani, aminek hatására maga lemond az összes birtokolt erőforrásáról. A folyamat csak akkor lesz újraindítva, ha minden erőforrás a rendelkezésére fog állni.

   A módszer hátránya, hogy fennáll annak a kockázata, hogy a folyamatot folytatni nem lehet, csak teljesen újraindítani, mivel az addigi eredményei, részeredményei elveszhetnek. A kiéheztetés ez esetben is elő fordulhat.

4. Körkörös várakozás

   Ez a feltétel is kizárható, amennyiben az erőforrásokat hierarchikusan megszámozzuk, és az erőforrások csak ebben a sorrendben foglalhatók le. Ezzel elérhetjük azt, hogy egy folyamat csak akkor kaphat meg egy adott sorszámú erőforrást, ha az adott erőforrásnál magasabb sorszámú erőforrást nem használ.

   A módszer hátránya, hogy a hierarchikus sorszám nem képes az igények változási dinamikáját követni, mivel statikus megoldás.

Mindezek ismeretében, akkor hogyan is kerülhető el a holtpont kialakulás, hatékony módon, miközben a holtpont kialakulásának feltételei továbbra is adottak. A megoldást az erőforrások kiosztásának módja képes biztosítani. A módszer akkor használható, ha:

- Biztos, hogy valamennyi olyan folyamat, amelynek erőforrás igényei maximálisan ki vannak elégítve, véges idő alatt lefut, s ezután felszabadulnak az általa lefoglalt erőforrások.
- Valamennyi folyamatról előre tudható, hogy milyen erőforrás típusból maximálisan mennyit igényel egyidejűleg lefoglalva.
- Az erőforrás igényeket csak akkor elégítjük ki, ha rendszer "biztonságos állapotban marad", azaz található olyan folyamat végrehajtási sorrend, melyben az egymást követő folyamatok maximális erőforrásigénye kielégíthető.

A biztonságos állapot tehát biztosan holtpont mentes. A nem biztonságos állapot még nem jelenti a holtpont automatikus kialakulását, de le kell szögezni, hogy a holtpont csak a nem biztonságos állapoton belül képes kialakulni.
A legkézenfekvőbb megoldás az úgynevezett bankár algoritmus.

2. Bankár Algoritmus

A Bankár algoritmust Edsger Wybe Dijkstra (1930-2002, holland matematikus-informatikus) dolgozta ki [az ő nevéhez fűződik a Szemafor, valamint a strukturált programozás fogalma is].
A legfontosabb alapelv az, hogy a Bank sohasem kölcsönzi ki egyszerre az összes pénzét, pláne nem egyetlen ügyfélnek, hanem ügyfelenkénti limitált hitelkereteket használ.
Az operációs rendszer csak azokat a folyamatokat engedi (egyszerre) futni, melyek erőforrásigénye (egyszerre) kielégíthető. A gyakorlatban sajnos a folyamatok nem mindig képesek pontosan előre jelezni az összes HW és SW erőforrás igényüket.
[Bankár algoritmus_v2.xls]

Az elvárható gondosság, mint fogalom, azt vélelmezi, hogy egy jól felépített rendszer megelőzési és elkerülési eljárásokat alkalmaz. Amennyiben ezek az eljárások hiányoznak, akkor csak eseti manuális módszereket lehet bevetni.