Operációs rendszerek TL;DR

1. Beugró fogalmak

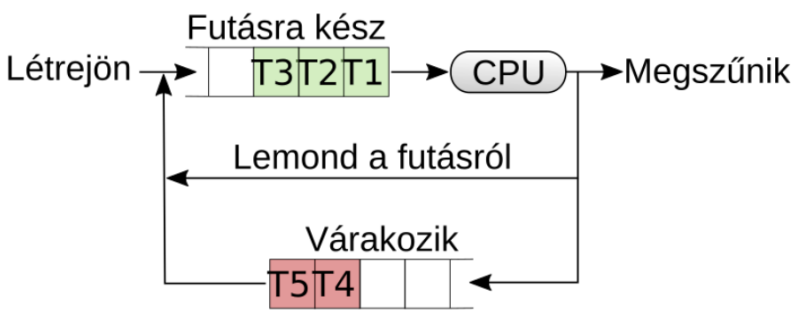
1. **operációs rendszer**: egy olyan szoftver, amelynek alapfeladata a felhasználói feladatok végrehajtásához szükséges környezet biztosítása, ezen túl számos adminisztratív funkcióval, gazdag eszköztámogatással és az informatikai megoldások széleskörű megvalósításával egészültek ki
2. **szabvány**: egy formális testület által hivatalos módon elfogadott specifikáció
3. **IEEE POSIX**: Institute of Electrical and Electronics Engineers - Portable Operating System Interface, a Unix-alapú rendszerek egységesítésére hozták létre, alapvetően programozói felületeket, parancsértelmezőt, az abban használható rendszerprogramokat, és az ezekhez szükséges rendszerelemeket szabványosítja
4. **kernel**: az egyetlen szoftver, ami mindig fut a számítógépen
5. **rendszerkönyvtár**: az operációs rendszer részét képező programkönyvtár, amelyet a programok felhasználnak működésük során
6. **rendszerprogram**: az operációs rendszer saját teendőit, például karbantartási és monitorozási feladatokat megoldó programok
7. **rendszerszolgáltatás**: olyan folyamatos működésű rendszerprogram, amely képes beérkező feladatok folyamatos megoldására
8. **rendszerhívás interfész**: a kernel része, egy programozói felület, amely a felhasználói módban működő programok számára nyújtott szolgáltatásait tartalmazza
9. **mikrokernel**: olyan kernel, amely csak az alapműködéséhez feltétlenül szükséges kódrészleteket tartalmazza, minden más funkciót felhasználói módban működtet
10. **monolitikus kernel**: olyan kernel, ahol a teljes kód és az adatok egyetlen címtérben helyezkednek el
11. **réteges kernel**: jól definiált, elhatárolható interfészekből áll
12. **moduláris kernel**: a rendszer programját részekre bontjuk, és működése során csak a szükséges részeket használjuk
13. **multiprogramozott OS**: több feladatot old meg egyszerre, a programokat háttértárról tölti be
14. **időosztásos OS**: a programokhoz időszeleteket rendelve váltakozva futtatja azokat
15. **beágyazott OS**: integrált funkcionalitású, komplex, olcsó csipeken fut
16. **valósidejű OS**: adott bemenetre meghatározott időn belül nagy valószínűséggel választ állít elő, ha a valószínűség 1, a rendszer kemény valósidejű
17. **ablakkezelő**: feladata az alkalmazásablakok elhelyezése és megjelenítése, testre szabható és bővíthető
18. **kijelzőszerver**: feladata a grafikus felhasználói felület elemeinek kiszolgálása erőforrásokkal: egységes interfész, GUI kérések és adatok feldolgozása és továbbítása a hardver felé, hardveresemények továbbítása az alkalmazás felé, jellemzően felhasználói módban
19. **ROM betöltő**: a gép bekapcsolásakor a processzor végrehajt egy fix címen kezdődő programot: hardverinicializálási feladatok (POST), és betöltőeszközök meghatározása
20. **RAM betöltő**: értelmezi a háttértár felépítését, megkeresi és betölti a következő szintű (OS) betöltőt
21. **OS betöltő**: már ismeri az OS-t (fájlrendszer felépítése, kernel betöltése), betölthet további programrészeket, lehet hozzá UI, betölti a kernelt és elindítja
22. **taszk**: egy végrehajtás alatt álló program
23. **absztrakt virtuális gép**: a kernel által a taszkok számára biztosított erőforrások számítógépként elképzelt együttese
24. **folyamat**: önálló memóriatartománnyal rendelkező taszk, amely szálakat futtathat
25. **szál**: egy szekvenciális működésű taszk, amely egy folyamaton belül más szálakkal közös memóriát használ
26. **kontextus**: egy program végrehajtási állapotának részletes leírója
27. **kontextusváltás**: az a művelet, amelynek során az éppen futó program kontextusa elmentődik, és egy másik program környezetét tölti be a processzor
28. **rövidtávú ütemezés**: avagy CPU-ütemezés, futásra kész állapotú taszkot választ, 1-100 ms, a kernel alapfeladata
29. **középtávú ütemezés**: taszkot választ (bármilyet), új állapotok: felfüggesztve vár / felfüggesztve futásra kész, percek/órák, a felhasználó és a kernel és kezdeményezheti
30. **hosszútávú ütemezés**: feladatot választ, taszkot indít, órák/napok/hetek…, nem a kernel hatásköre
31. **preemptív ütemező**: saját döntése alapján futó állapotból futásra kész állapotba helyezhet át taszkokat
32. **kooperatív ütemező**: saját döntése alapján futó állapotból nem helyez át futásra kész állapotba egyetlen taszkot sem
33. **körülfordulási idő**: a taszk létrejöttétől a megszűnéséig eltelt teljes idő
34. **várakozási idő**: a taszk összes nem futó állapotában eltöltött ideje
35. **válaszidő**: az az idő, amely egy esemény bekövetkezése és a taszk arra adott válasza között telik el
36. **CPU kihasználtság**: annak az időnek az összes időhöz viszonyított százalékos mértéke, amikor a processzor taszkok utasításait hajtja végre
37. **átbocsájtó képesség**: az időegység alatt elvégzett taszkok száma
38. **rezsiköltség**: az ütemező programjának végrehajtására fordított processzoridő
39. **CPU-löket**: a taszk processzoron végrehajtott utasításainak sorozata
40. **I/O-löket**: a taszk működésének azon fázisa, amikor I/O művelet végrehajtására vár
41. **konvoj-hatás**: az a jelenség, amikor kooperatív ütemezés esetén egy nagy CPU-löketidejű futó taszk mögött feltorlódnak a futásra kész taszkok
42. **prioritás**: a taszk jellemzője, amely végrehajtásának fontosságát fejezi ki
43. **külső prioritás**: a taszk operációs rendszeren kívüli meghatározott végrehajtási fontossága
44. **belső prioritás**: a taszk operációs rendszer által meghatározott végrehajtási fontossága
45. **statikus prioritás**: a taszk elindulása előtt meghatározott, és megszűnéséig érvényes prioritásérték
46. **dinamikus prioritás**: a taszk életciklusa alatt kiszámított prioritásérték
47. **kiéheztetés**: az a jelenség, amikor prioritásos ütemezés esetén egy taszkot folyamatosan megelőznek nála magasabb prioritásúak, ezért nem jut processzorhoz
48. **öregítés**: a futásra kész taszk prioritásának növelése az ebben az állapotban eltöltött idejével arányosan
49. **többszintű ütemező**: a futásra kész taszkokat több, különálló halmazban tartja nyilván, a halmazok és azok elemei közül különböző algoritmusok szerint választja ki a futó állapotba kerülő taszkot
50. **statikus többszintű ütemező**: az ütemezési halmazok között nem helyez át taszkokat
51. **dinamikus többszintű ütemező**: a taszkok ütemezési halmazokhoz rendelését működése során megválaszthatja
52. **CPU-affinitás**: a taszk és az azt végrehajtó processzor kötődése: laza - megpróbálja, de nem garantált (ma jellemző) / kemény - garantált taszk - CPU párosítás, rendszerhívással és a felhasználói felületen állítható be, processzorhalmazzal is lehet
53. **memória affinitás**: CPU - memória elemek között hardveres kapcsolat (affinitás) van, a kernel transzparens módon kezeli, az allokáció a végrehajtó CPU memóriatartományában történik
54. **címleképezés**: a virtuális és fizikai címek megfeleltetése hardveres támogatással
55. **lapozás**: a taszkok memóriatartományainak részekre bontása hardveres támogatással
56. **cserehely**: a gyors fizikai memória kapacitásának kiterjesztése lassú háttértárolókkal
57. **MMU**: virtuális (CPU teljes címtartománya) és fizikai (memória + háttértáron allokált) címeket rendel össze

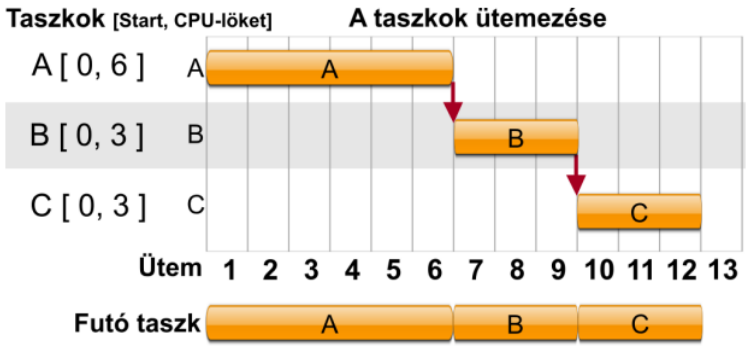
58. **lapszervezésű virtuális memória-kezelés**: virtuális címtartomány: lapok, fizikai memória: keretek, háttértár: blokkok, laptábla: lapok ← → keretek, Translation Lookaside Buffer (TLB): címfordító gyorstár
59. **laptábla**: lapok és keretek összerendelését tartalmazza
60. **kerettábla**: a kereteket tartalmazza, sorszámmal indexelve, állapottal (szabad, foglalt, DMA alatt), és hivatkozásszámlálóval (hány taszk használja)
61. **diszk blokk leíró**: tartalma: háttértár eszközazonosító, blokk sorszám, típus (swap/zerofill/fill-from-text)
62. **tárcsere**: taszkok teljes memóriatartományának háttértárra írása
63. **laphiba**: a hivatkozott lap nincs a fizikai memóriában
64. **védelmi hiba**: hibás címzés (érvénytelen cím, hozzáférési hiba)
65. **copy-on-write technika**: csak akkor hoz létre valódi másolatot, ha abba írás történne
66. **vergődés**: gyakori laphibák miatt a teljesítmény jelentősen romlik, a taszkok számának kordában tartásával (középtávú ütemezéssel) kezelhető
67. **igény szerinti lapozás**: nem próbál meg jósolni, csak a jelenlegi igényekkel foglalkozik - csak laphiba esetén fut és a szükséges lapot hozza
68. **előretekintő lapozás**: több lapot hoz be egyszerre, megpróbálja kitalálni, mit kell lapozni (lokalitási jellemzők, laphibák a múltból)
69. **lapcsere algoritmus**: kiválaszt egy felszabadítandó keretet, amit kiír a cserehelyre
70. **lapok tárba fagyasztása**: a frissen behozott lapok múlt nélkül könnyen cserére jelölhetők, pedig meg kellene tartani őket, és az I/O-művelet alatt álló lapokat se kellene módosítani
71. **laplopó taszk**: feladata üres keretek biztosítása, a szabad keretek számát igyekszik két érték között tartani, ha egy minimum mennyiségű üres keret nincs meg, elkezd kereteket lopni
72. **PRAM modell**: közösen használható memória, az egyidejű műveleteket nem keveri, szinkron műveletek, pl. szálak vagy osztott memória
73. **direkt kommunikáció**: a küldő taszk az adatátviteli rendszeren keresztül közvetlenül juttat el adatot a fogadóhoz
74. **indirekt kommunikáció**: a küldő taszk az adatátviteli rendszer postaládáján keresztül juttat el adatot a fogadóhoz
75. **szinkron adatátvitel**: a küldés és fogadás blokkoló művelet, ettől a taszk várakozó állapotba kerül
76. **aszinkron adatátvitel**: nincs blokkolás, a taszk fut tovább, de az eredmény még nem érhető el, a nem kézbesített üzeneteket tárolni kell
77. **jelzés**: értesítés eseményről, taszk → taszk vagy kernel → taszk
78. **távoli eljáráshívás és részei**: egy másik taszk programjában lévő eljárás hívása, aszimmetrikus szerver-kliens kommunikáció, hálózatra épül, adatkonverziót is végez
79. **postaláda**: véges számú (sokszor csak egy) üzenetet tárol, korlátos méretben, a címzett helyett a postaláda címezhető
80. **üzenetsor**: végtelen, de korlátos méretű üzenetet tárol, jellemzően szűrni is lehet, működhet rendszerek közt is
81. **csővezeték**: végtelen, folytonos adatküldésre és -fogadásra, egyszerre több vevője is lehet
82. **szinkronizáció**: a taszkok működésének összehangolása a művelet-végrehajtás időbeli korlátozásával
83. **kölcsönös kizárás**: taszkok egymást kizáró működése, erőforrás-védelem, versenyhelyzetek kezelése, pl. közös erőforrások használata
84. **kritikus szakasz**: utasítások egy olyan sorozata, amelyet egy időben a taszkoknak csak egy korlátozott halmaza hajthat végre
85. **szemafor**: atomi műveletekkel rendelkező változó (várakozás (P) és továbbengedés (V))
86. **mutex**: a kritikus szakasz védelmére alkalmazott zárolási eszköz, bináris szemafor
87. **spinlock**: olyan lock, mutex, vagy szemafor, ami aktívan várakozik, rövid kritikus szakaszok esetére
88. **readerwriterlock**: tetszőleges számú olvasó, de ha író jön, akkor blokkolódik, amíg minden olvasó ki nem lép
89. **holtpont és kialakulásának feltételei**: taszkok egy H halmazában található valamennyi taszk olyan eseményre vár, amelyet csak H-n belüli taszkok idézhetnek elő
90. **erőforrás-allokációs gráf**: erőforrás foglalási gráf, csúcsai a szálak és az erőforrások, él megy az erőforrásba, ha a szál lefoglalja, és él indul visszafelé, ha feloldja, kör esetén holtpont van
91. **optimista zárolás**: nem zárol, de detektálja, és korrigálja a holtpontot, kevés konfliktus esetén gyorsít, sok konfliktus esetén lassít
92. **fájl**: az adattárolás logikai egysége, névvel hivatkozható, egyes rendszereken kiterjesztése is van
93. **könyvtár**: fájlok szervezésének logikai egysége, egy gyűjtőhely, fájlokat és más könyvtárakat tartalmazhat
94. **kötet**: fájlok és könyvtárak összefüggő halmaza, logikai tárolási egysége, fizikai tárolási egységhez egyértelműen hozzárendelhető
95. **fájlrendszer**: fájlok és könyvtárak fizikai tárolási egysége és szervezési rendszere
96. **partíció**: a háttértár szervezési egysége, amely egy fájlrendszer tárolására képes
97. **POSIX jogosultságok**: 3x3 bit, tulaj/csoport/mások x olvasás/írás/futtatás, könyvtárnál olvasás és futtatás is kell a listázáshoz
98. **csatolás**: a fizikai tárolóhelyet a logikai struktúra adott pontjához rendeli
99. **csatlakozási pont**: a logikai struktúra azon eleme, amit a csatolt fájlrendszer elfed
100. **ajánlott fájlzárolás**: OS ad hozzá eszközt, amelyik taszk nem ezeket használja, nem vesz részt benne
101. **kötelező fájlzárolás**: kernel mechanizmusok biztosítják, a rendszerhívások figyelembe veszik, így minden taszk számára kötelező
102. **mmap()**: fájlok megosztott elérése memórián keresztül, többszörös hozzáférés, konzisztencia, és kizárás kezelésével
103. **szuperblokk**: fájlrendszer metaadatok, a lemez minden cilindercsoportjába bemásolva
104. **inode**: fájl metaadatok a fájlrendszerben
105. **naplózó fájlrendszer**: a változások naplóba íródnak, ami mindig a diszken van, a metaadatokon végzett műveletek tranzakcióként mennek végbe, a tranzakció a naplóba írás után zárul, a napló egy szekvenciálisan írható körpuffer, ha a műveletet a fájlrendszer is végrehajtja, kikerül a naplóból, összeomló rendszer induláskor feldolgozza a naplót
106. **LVM**: virtuális tárolórendszerek logikai kötetkezelése, a fizikai eszközök határain átnyúló allokációs rendszer, a partícióknál rugalmasabb helygazdálkodás, építőelemei diszkek és partíciók
107. **RAID 0**: csíkozás, az adatot egyenletesen szétteríti a diszkeken
108. **RAID 1**: tükrözés, az adatokat többszörözve tárolja
109. **RAID 01**: 0 felett 1, csíkozott diszkek tükrözése
110. **RAID 10**: 1 felett 0, tükrözött adatblokkokat csíkozva terít el, gyorsabb és megbízhatóbb
111. **RAID 5**: blokkszintű csíkozása n lemeznek egy paritással, paritásblokkok egyenletesen elhelyezve, egy diszk hibája ellen véd, néma hiba lehetősége: a kiesett diszk visszaállítása közben további hiba lehetséges
112. **RAID 6**: RAID 5 + egy újabb paritáslemez, javított megbízhatóság, még egy diszk kieshet, ez a javasolt
113. **NAS**: hálózati fájltároló, Network Attached Storage, magas szintű, fájlorientált átvitelt valósít meg
114. **SAN**: adatblokk tárolást megvalósító Storage Area Network, alacsony szintű átvitelt valósít meg, távoli eszközként használható
115. **NFS**: Network File System
116. **iSCSI**: internet SCSI, IP-alapú SCSI parancsok átvitelére szolgál
117. **virtualizáció**: erőforrás virtuális (szoftveres) változatának létrehozása, amely az eredetire támaszkodva, ahhoz hasonlóan, de attól elválasztott módon működik
118. **gazda gép**: a virtualizált erőforrást biztosítja
119. **vendég gép**: az erőforrás felhasználója
120. **virtuális gép monitor**: a virtuális gépeket felügyelő program

121. **bare metal virtualizáció**: a hardvert a VMM kezeli, a gazda gépen más nem fut, ilyenkor a VMM neve hypervisor, natív virtualizáció, ha hardveres a támogatás, és paravirtualizáció, ha a fizikai hardverhez hasonló virtuális hardverrel fut
122. **IaaS**: infrastructure as a service, teljes hardver felhőn keresztüli bérbeadása (Azure, Amazon EC3, Linode)
123. **PaaS**: platform as a service, futtatókörnyezetet nyújtó szolgáltatások (Azure, Google AppEngine, Heroku)
124. **SaaS**: software as a service, szoftverszolgáltatás, pl. adatbáziskezelő vagy szövegszerkesztő (Google Docs, Office365)
125. **védett végrehajtási mód**: bármi megtehető, fennhatóságot gyakorol minden program felett, szabályozza a működésüket (pl. start/stop)
126. **felhasználói végrehajtási mód**: hardveresen betartatott korlátozásai vannak
127. **virtuális fájlrendszer**: egy implementáció-független absztrakció a sokféle fájlrendszer kezelésére, többféle fs egyidejű támogatása, egységes kezelés a csatlakoztatás után, speciális fájlrendszerek uniform megvalósítása, modulárisan bővíthető, modern UNIX rendszerek alapja
128. **láncolt listás adatblokk tárolás**: a fájlok adatait kisebb részekre bontja, az egyes részek hivatkoznak a további részekre, az első rész címe a metaadatban van, a következő blokké minden blokkban, soros elérésre hatékony, érzékeny a hibákra, az üres blokkokat is lehet így tárolni
129. **indexelt adatblokk tárolás**: a fájlokat blokkokra bontja, a blokkok helyéről indexet készít, a túl nagy index láncolt listásan tárolható
130. **többszörös indexelt adatblokk címtábla**: adatblokkokból direkt blokk címes index, azokból indirekt blokk címes index

1.1. Egyszerű ütemezési algoritmusok

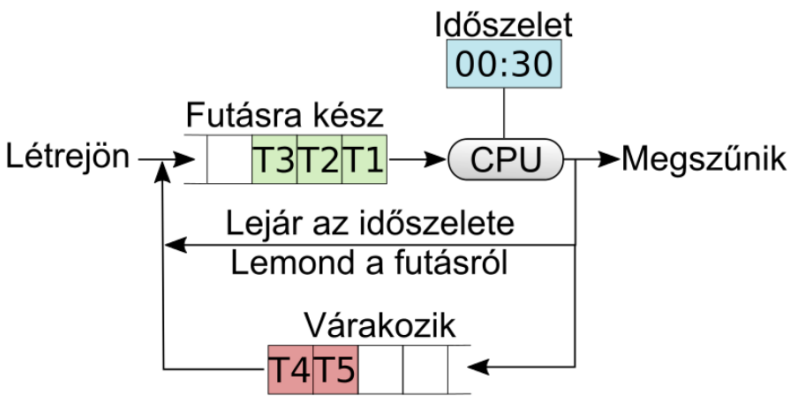
Legrégebben várakozó (first come first served, FCFS)

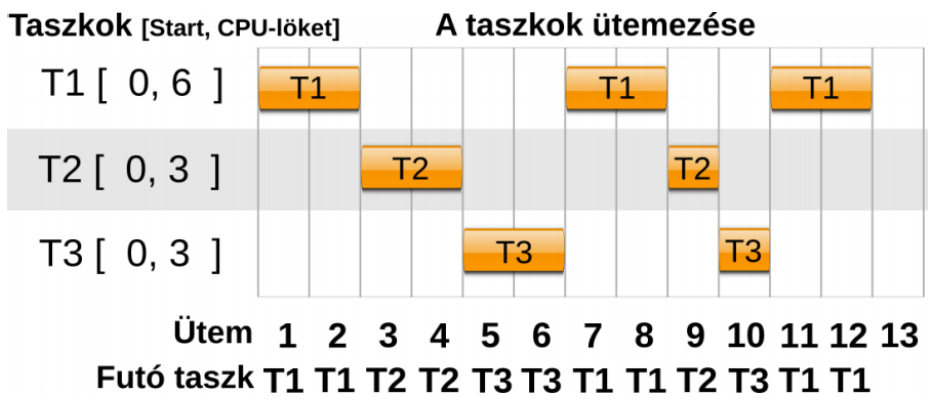




Kooperatív, egyszerű adatstruktúra és algoritmus (FIFO), O(1) komplexitás, minimális rezsiköltség, de egy nagy löketidejű taszk mögött feltorlódik a többi (konvoj-hatás), ez jellemzően magas válaszidőt eredményez. Javításai: vágjuk el a hosszú taszkokat (körforgó), vagy rendezzük a sort (legrövidebb löketidejű ütemezés).

Körforgó ütemezés (round-robin, RR)

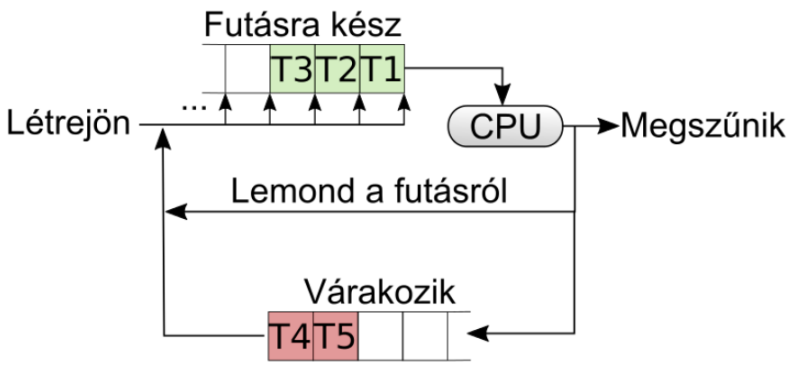


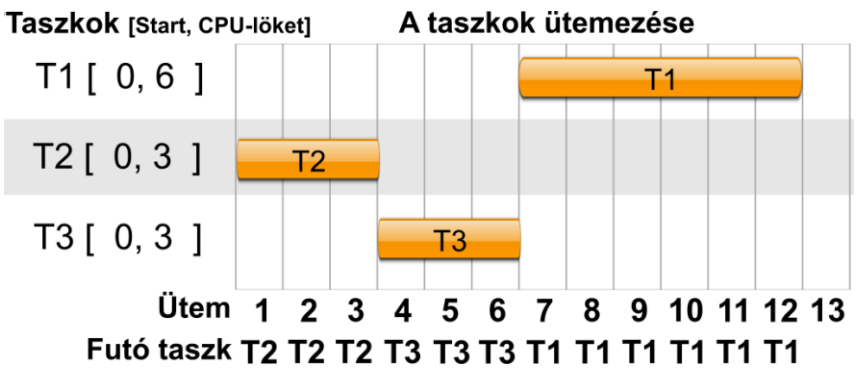


Preemptív, egyszerű adatstruktúra és algoritmus (FIFO), O(1) komplexitás, minimális rezsiköltség jól megválasztott időszelettel, túl nagy szelettel FCFS, túl kicsivel nagy a rezsiköltség.

Legrövidebb löketidejú (shortest job first, SJF)

A futásra kész taszkok löketidő szerint sorban vannak.





Kooperatív, bonyolultabb adatstruktúra (löketidő szerinti sor), O(N) komplexitás a keresés miatt, a rezsiköltség nagyobb, mint FCFS vagy RR esetében, a löketidőt nem ismerjük, így becsült.

Legrövidebb hátralévő idejű (shortest remaining time first, SRTF)

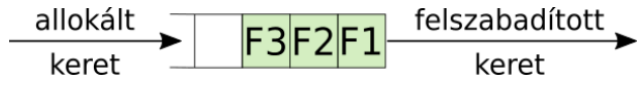
Az SJF preemptív változata, ha új taszk lesz futásra kész, és az gyorsabban kész a jelenleg futónál, akkor futni fog.

Prioritásos ütemező

A legmagasabb prioritású taszkot választja. A prioritás lehet belső (OS által meghatározott) vagy külső (taszk vagy felhasználó által meghatározott). Lehet még statikus (taszk futása előtt meghatározott) vagy dinamikus (futás közben változó) is a prioritás.

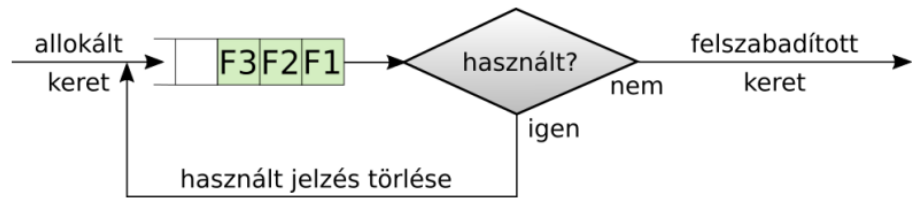
1.2. Lapcsere algoritmusok

FIFO



Egyszerű algoritmus és adatstruktúra (FIFO), előrenéző (nem érdekli a múlt), O(1) komplexitás, minimális rezsiköltség, könnyű megvalósítani, a lapok jövőbeni használatát rosszul becsli, nem figyeli a módosítást.

Újabb esély (second chance, SC)

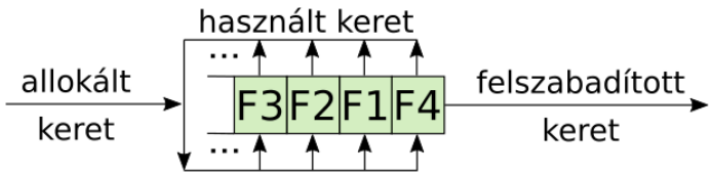


Egyszerű algoritmus és adatstruktúra (FIFO), hátranéző, a használt lapokat nem szabadítja fel, O(1) komplexitás, minimális rezsiköltség, könnyű megvalósítani, jobban becsüli a jövőbeni használatot a FIFO-nál, nem figyeli a módosítást, állandóan mozgatja a taszkokat a FIFO-ban.

Óra

Ugyanaz, mint az újabb esély, csak körkörösen mozog egy listában, így jobban becsül, és nem mozgatja a taszkokat.

Legrégebben nem használt (least recently used, LRU)



Lapok rendezése használati idő szerint, pl. láncolt listával, bonyolultabb algoritmus és struktúra, hátranéző, O(N), hardveres támogatással kis rezsiköltség, egyébként magas. A frissen behozott lapokat nagy eséllyel kidobja, így kell tárba fagyasztás.