**ZÁRTHELYI 1. ALGORITMUSOK**

**I. Holtpont**

*I/1. Biztonságos sorozat keresés (amely alapja, hogy a folyamat csak maximális erőforrás megléte esetén fut csak le)*

Adott egy rendszer, amelyben 1 erőforrás osztály van, 10 erőforrással. A rendszerben 3 folyamat (P1, P2, P3) található, amelyek az alábbi foglalásokkal F=(3, 2, 2) és maximális igényekkel M=(9, 4, 7) rendelkeznek. Ha úgy véli, hogy a rendszer biztonságos állapotban van, akkor adja meg a folyamatoknak egy biztonságos sorrendjét, különben jelölje az összeset „-„ jellel:

Biztonságos sorozat:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 🡪 |  | 🡪 |  |

1. lépés: Elkészítem a kiindulási táblázatot a megadott adatok alapján. Szükségünk van egy Foglal, egy Max, és egy Még oszlopra. Mivel a rendszerben 3 folyamat található meg (P1, P2, P3), így 3 sorra. A foglalásokat F=(3, 2, 2) a Foglal oszlopba, a maximális igényeket M=(9, 4, 7) a Max oszlopba írom be. A Még oszlopot kitöltöm a Max és a Foglal oszlopok értékeinek különbsége alapján. Utolsó lépésként összeadom a foglalt erőforrások számát, majd megnézem, hogy hány szabad maradt a 10 erőforrásból. Ezt az értéket a Szabad sorba írom be.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Foglal | Max | Még |
| P1 | 3 | 9 | 6 |
| P2 | 2 | 4 | 2 |
| P3 | 2 | 7 | 5 |
| Szabad: 3 | | | |

2. lépés: A szabad erőforrások száma alapján meghatározom, hogy melyik folyamat erőforrás igényét tudom maximálisan kielégíteni. A választásom a P2 folyamatra esett mivel neki még 2 szabad erőforrásra van szüksége. Ezt követően a foglalt erőforrások számát (Foglal) a maximum értékre, a kérhető erőforrások számát (Még) pedig 0 értékre állítom. Végül összeadom a foglalt erőforrásokat, és kivonom a rendszerben lévő erőforrásokat számából: 10-9=1. Az szabad erőforrások száma 1 lesz.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Foglal | Max | Még |
| P1 | 3 | 9 | 6 |
| P2 | 4 | 4 | 0 |
| P3 | 2 | 7 | 5 |
| Szabad: 1 | | | |

3. lépés: Lefut a P2 folyamat és visszaadja az általa foglalt erőforrásokat, így a szabad erdőforrások száma 5 lesz.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Foglal | Max | Még |
| P1 | 3 | 9 | 6 |
| P2 | - | - | - |
| P3 | 2 | 7 | 5 |
| Szabad: 5 | | | |

4. lépés: A szabad erőforrások száma alapján újra meghatározom, hogy melyik folyamat erőforrás igényét tudom maximálisan kielégíteni. A választásom a P3 folyamatra esett mivel neki még 5 szabad erőforrásra van szüksége. Ezt követően a foglalt erőforrások számát (Foglal) a maximum értékre, a kérhető erőforrások számát (Még) pedig 0 értékre állítom. Végül összeadom a foglalt erőforrásokat, és kivonom a rendszerben lévő erőforrásokat számából: 10-10=0. Az szabad erőforrások száma 0 lesz.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Foglal | Max | Még |
| P1 | 3 | 9 | 6 |
| P2 | - | - | - |
| P3 | 7 | 7 | 0 |
| Szabad: 0 | | | |

5. lépés: Lefut a P3 folyamat és visszaadja az általa foglalt erőforrásokat, így a szabad erdőforrások száma 7 lesz.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Foglal | Max | Még |
| P1 | 3 | 9 | 6 |
| P2 | - | - | - |
| P3 | - | - | - |
| Szabad: 7 | | | |

6. lépés: A szabad erőforrások száma alapján újra meghatározom, hogy melyik folyamat erőforrás igényét tudom maximálisan kielégíteni. A választásom a P1 folyamatra esett mivel neki még 6 szabad erőforrásra van szüksége. Ezt követően a foglalt erőforrások számát (Foglal) a maximum értékre, a kérhető erőforrások számát (Még) pedig 0 értékre állítom. Végül összeadom a foglalt erőforrásokat, és kivonom a rendszerben lévő erőforrásokat számából: 10-10=0. Az szabad erőforrások száma 0 lesz.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Foglal | Max | Még |
| P1 | 9 | 9 | 0 |
| P2 | - | - | - |
| P3 | - | - | - |
| Szabad: 1 | | | |

7. lépés: Lefut a P1 folyamat és visszaadja az általa foglalt erőforrásokat, így a szabad erdőforrások száma 10 lesz.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Foglal | Max | Még |
| P1 | - | - | - |
| P2 | - | - | - |
| P3 | - | - | - |
| Szabad: 10 | | | |

8. lépés: Mivel mindegyik folyamat erőforrásigényét teljesíteni lehetett, így meg lehet határozni egy biztonságos sorozatot:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P2 | 🡪 | P3 | 🡪 | P1 |

*I/2. Coffman algoritmus*

A Coffman algoritmus segítségével állapítsa meg, hogy van-e holtpont, ha a rendszerben négy folyamatunk van (P1, P2, P3 és P4), amelyek három erőforrásosztályból már foglalnak néhány erőforrást, de továbbiakat is igényelnek a következő táblázat szerint:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Foglal | | | Kér | | | Szabad | | |
|  | A | B | C | A | B | C | A | B | C |
| P1 | 1 | 4 | 0 | 0 | 0 | 5 | 5 | 1 | 0 |
| P2 | 5 | 1 | 3 | 2 | 0 | 2 |  |  |  |
| P3 | 1 | 0 | 3 | 0 | 2 | 0 |  |  |  |
| P4 | 2 | 4 | 1 | 4 | 0 | 0 |  |  |  |

Ha úgy véli, hogy a rendszerben nincs holtpont, akkor adja meg a folyamatoknak egy biztonságos sorozatát, különben jelölje az összeset „-„ jellel:

Biztonságos sorozat:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 🡪 |  | 🡪 |  | 🡪 |  |

1. lépés: Olyan folyamatot keresek, amely erőforrásigényét teljesíteni tudom a szabad erőforrások (A:5, B:1, C:0) függvényében. A választásom a P4 folyamatra esett, mivel az ő erőforrásigénye (A:4, B:0, C:0) teljesíthető.

2. lépés: Kivonom a szabad erőforrások számából a P4 folyamat erőforrás igényét. Majd ezt követően a P4 folyamat által foglalt erőforrásokat hozzáadom a szabad erőforrásokhoz.

510 – 400 = 110, 110 + 241 = 351

Az új szabad erőforrások száma a következő: A:3, B:5, C:1

3. lépés: A P4 által kért erőforrásokat áthelyezem a Foglal oszlopba. Mivel a P4 nem kér több erőforrást a lefutáshoz, így a P4-hez tartozó Kér oszlop mezői kihúzhatók.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Foglal | | | Kér | | | Szabad | | |
|  | A | B | C | A | B | C | A | B | C |
| P1 | 1 | 4 | 0 | 0 | 0 | 5 | 3 | 5 | 1 |
| P2 | 5 | 1 | 3 | 2 | 0 | 2 |  |  |  |
| P3 | 1 | 0 | 3 | 0 | 2 | 0 |  |  |  |
| P4 | 4 | 0 | 0 | - | - | - |  |  |  |

4. lépés: Újra megvizsgálom, hogy tudok-e találni olyan folyamatot, amely erőforrásigényét teljesíteni tudom a szabad erőforrások (A:3, B:5, C:1) függvényében. A választásom a P3 folyamatra esett, mivel az ő erőforrásigénye (A:0, B:2, C:0) teljesíthető.

5. lépés: Kivonom a szabad erőforrások számából a P3 folyamat erőforrás igényét. Majd ezt követően a P3 folyamat által foglalt erőforrásokat hozzáadom a szabad erőforrásokhoz.

351 – 020 = 331, 331 + 103 = 434

Az új szabad erőforrások száma a következő: A:4, B:3, C:4

6. lépés: A P3 által kért erőforrásokat áthelyezem a Foglal oszlopba. Mivel a P3 nem kér több erőforrást a lefutáshoz, így a P3-hez tartozó Kér oszlop mezői kihúzhatók.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Foglal | | | Kér | | | Szabad | | |
|  | A | B | C | A | B | C | A | B | C |
| P1 | 1 | 4 | 0 | 0 | 0 | 5 | 4 | 3 | 4 |
| P2 | 5 | 1 | 3 | 2 | 0 | 2 |  |  |  |
| P3 | 0 | 2 | 0 | - | - | - |  |  |  |
| P4 | 4 | 0 | 0 | - | - | - |  |  |  |

7. lépés: Újra megvizsgálom, hogy tudok-e találni olyan folyamatot, amely erőforrásigényét teljesíteni tudom a szabad erőforrások (A:4, B:3, C:4) függvényében. A választásom a P2 folyamatra esett, mivel az ő erőforrásigénye (A:2, B:0, C:2) teljesíthető.

8. lépés: Kivonom a szabad erőforrások számából a P2 folyamat erőforrás igényét. Majd ezt követően a P2 folyamat által foglalt erőforrásokat hozzáadom a szabad erőforrásokhoz.

434 – 202 = 232, 232 + 513 = 745

Az új szabad erőforrások száma a következő: A:7, B:4, C:5

9. lépés: A P2 által kért erőforrásokat áthelyezem a Foglal oszlopba. Mivel a P2 nem kér több erőforrást a lefutáshoz, így a P2-hez tartozó Kér oszlop mezői kihúzhatók.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Foglal | | | Kér | | | Szabad | | |
|  | A | B | C | A | B | C | A | B | C |
| P1 | 1 | 4 | 0 | 0 | 0 | 5 | 7 | 4 | 5 |
| P2 | 2 | 0 | 2 | - | - | - |  |  |  |
| P3 | 0 | 2 | 0 | - | - | - |  |  |  |
| P4 | 4 | 0 | 0 | - | - | - |  |  |  |

10. lépés: Újra megvizsgálom, hogy tudok-e találni olyan folyamatot, amely erőforrásigényét teljesíteni tudom a szabad erőforrások (A:7, B:4, C:5) függvényében. A választásom a P1 folyamatra esett, mivel az ő erőforrásigénye (A:0, B:0, C:5) teljesíthető.

11. lépés: Kivonom a szabad erőforrások számából a P1 folyamat erőforrás igényét. Majd ezt követően a P1 folyamat által foglalt erőforrásokat hozzáadom a szabad erőforrásokhoz.

745 – 005 = 740, 740 + 140 = 880

Az új szabad erőforrások száma a következő: A:8, B:8, C:0

12. lépés: A P1 által kért erőforrásokat áthelyezem a Foglal oszlopba. Mivel a P1 nem kér több erőforrást a lefutáshoz, így a P1-hez tartozó Kér oszlop mezői kihúzhatók.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Foglal | | | Kér | | | Szabad | | |
|  | A | B | C | A | B | C | A | B | C |
| P1 | 0 | 0 | 5 | - | - | - | 8 | 8 | 0 |
| P2 | 2 | 0 | 2 | - | - | - |  |  |  |
| P3 | 0 | 2 | 0 | - | - | - |  |  |  |
| P4 | 4 | 0 | 0 | - | - | - |  |  |  |

13. lépés: Mivel mindegyik folyamat erőforrásigényét teljesíteni lehetett, így meg lehet határozni egy biztonságos sorozatot:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| P4 | 🡪 | P3 | 🡪 | P2 | 🡪 | P1 |

**II. Ütemezés**

*II/1. Legrégebben várakozó (First Come First Served, FCFS)*

*Adott öt folyamat P=(1, 2, 3, 4, 5), amelyeknek futásideje C=(1, 2, 6, 5, 3). A folyamatok T=(2, 4, 5, 0, 8) időpontban kerülnek futásra kész állapotba. A rendszerben legrégebben várakozó (FCFS) CPU ütemezés működik.*

*Adja meg a folyamatok ütemezését:*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 🡪 |  | 🡪 |  | 🡪 |  | 🡪 |  |

1. lépés: Elkészítem a kiindulási táblázatot a megadott adatok alapján. A folyamatokat az 1. oszlopba, a folyamatok futásra állapotainak idejét (T) a 2. oszlopba, még a futási időt (C) a harmadik oszlopba írom be.

|  |  |  |
| --- | --- | --- |
|  | Érkezik | CPU-idő |
| P1 | 2 | 1 |
| P2 | 3 | 2 |
| P3 | 4 | 6 |
| P4 | 0 | 5 |
| P5 | 8 | 3 |

2. lépés: Az algoritmus alapja, hogy az a folyamat lesz leghamarabb kiszolgálva, amelyik leghamarabb érkezett meg futásra kész állapotba. Érkezési sorrendben a folyamatok ütemezése a következő:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P4 | 🡪 | P1 | 🡪 | P2 | 🡪 | P3 | 🡪 | P5 |

*II/2. Körbeforgó (Round Robin, RR)*

Adott négy folyamat P=(1, 2, 3, 4), amelyeknek futásideje C=(3, 6, 2, 5). A folyamatok T=(1, 3, 0, 5) időpontban kerülnek futásra kész állapotba. A rendszerben körforgó (RR) CPU ütemezés működik, az időszelet hossza 3 ms.

Adja meg a folyamatok ütemezését:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 🡪 |  | 🡪 |  | 🡪 |  | 🡪 |  | 🡪 |  | 🡪 |  | 🡪 |  |

(Ha valamelyik helyre már nincs mit ütemezni, akkor jelölje "-" jellel!)

1. lépés: Elkészítem a kiindulási táblázatot a megadott adatok alapján. A folyamatokat az 1. oszlopba, a folyamatok futásra állapotainak idejét (T) a 2. oszlopba, még a futási időt (C) a harmadik oszlopba írom be.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Érkezik | CPU-idő | Indul | Vége | Vár |
| P1 | 1 | 3 |  |  |  |
| P2 | 3 | 6 |  |  |  |
| P3 | 0 | 2 |  |  |  |
| P4 | 5 | 5 |  |  |  |

2. lépés: Megkeresem a legkorábban érkező folyamatot (P3), majd az érkezési idejét, beírom az Indul oszlop megfelelő helyére. Mivel a P3 folyamat CPU-ideje (2) kisebb, mint az időszelet hossza (3), így az indulási időhöz hozzáadom a CPU-időt, majd ezt az értéket a Vége oszlop megfelelő sorába írom be. Végül, mivel a folyamatnak nem kellett várakoznia, így a Vár oszlop megfelelő sorába a 0 értéket írom be.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Érkezik | CPU-idő | Indul | Vége | Vár |
| P1 | 1 | 3 |  |  |  |
| P2 | 3 | 6 |  |  |  |
| P3 | 0 | 2 | 0 | 2 | 0 |
| P4 | 5 | 5 |  |  |  |

3. lépés: Megkeresem a következő legkorábban érkező folyamatot (P1), majd az indulási idejének megadom az előző folyamat (P3) végzési idejét. Mivel a P1 folyamat CPU-ideje (3) megegyezik az időszelet hosszával (3), így az indulási időhöz hozzáadom a CPU-időt, majd ezt az értéket (5) a Vége oszlop megfelelő sorába írom be. Végül, mivel a P1 folyamat az 1. időszeletben érkezett, de csak a 2. időszeletben indult, így 1 időszeletet kellett várnia. Ezt az értéket a Vár oszlop megfelelő helyére írom be.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Érkezik | CPU-idő | Indul | Vége | Vár |
| P1 | 1 | 3 | 2 | 5 | 1 |
| P2 | 3 | 6 |  |  |  |
| P3 | 0 | 2 | 0 | 2 | 0 |
| P4 | 5 | 5 |  |  |  |

4. lépés: Megkeresem a következő legkorábban érkező folyamatot (P2), majd az indulási idejének megadom az előző folyamat (P1) végzési idejét. Mivel a P2 folyamat CPU-ideje (6) nagyobb, mint az időszelet hossza (3), így az indulási időhöz (5) hozzáadom az időszelet hosszát (3), majd ezt az értéket (8) írom be a végzési időnek. Ezt követően létrehozok egy új P2 folyamatot, amelynek érkezési ideje, az előző végzési ideje (8), a CPU-ideje pedig az eredeti CPU-idő és az időszelet különbsége (6-3=3). Végül, mivel a P2 folyamat a 3. időszeletben érkezett, de csak az 5. időszeletben indult el, így 2 időszeletet kellett várnia. Ezt az értéket a Vár oszlop megfelelő helyére írom be.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Érkezik | CPU-idő | Indul | Vége | Vár |
| P1 | 1 | 3 | 2 | 5 | 1 |
| P2 | 3 | 6 | 5 | 8 | 2 |
| P3 | 0 | 2 | 0 | 2 | 0 |
| P4 | 5 | 5 |  |  |  |
| P2 | 8 | 3 |  |  |  |

5. lépés: Megkeresem a következő legkorábban érkező folyamatot (P4), majd az indulási idejének megadom az előző folyamat (P2) végzési idejét. Mivel a P4 folyamat CPU-ideje (5) nagyobb, mint az időszelet hossza (3), így az indulási időhöz (8) hozzáadom az időszelet hosszát (3), majd ezt az értéket (11) írom be a végzési időnek. Ezt követően létrehozok egy új P4 folyamatot, amelynek érkezési ideje, az előző végzési ideje (11), a CPU-ideje pedig az eredeti CPU-idő és az időszelet különbsége (5-3=2). Végül, mivel a P4 folyamat az 5. időszeletben érkezett, de csak a 8. időszeletben indult el, így 3 időszeletet kellett várnia. Ezt az értéket a Vár oszlop megfelelő helyére írom be.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Érkezik | CPU-idő | Indul | Vége | Vár |
| P1 | 1 | 3 | 2 | 5 | 1 |
| P2 | 3 | 6 | 5 | 8 | 2 |
| P3 | 0 | 2 | 0 | 2 | 0 |
| P4 | 5 | 5 | 8 | 11 | 3 |
| P2 | 8 | 3 |  |  |  |
| P4 | 11 | 2 |  |  |  |

6. lépés: Megkeresem a következő legkorábban érkező folyamatot (P2), majd az indulási idejének megadom az előző folyamat (P4) végzési idejét. Mivel a P2 folyamat CPU-ideje (3) megegyezik az időszelet hosszával (3), így az indulási időhöz hozzáadom a CPU-időt, majd ezt az értéket (14) a Vége oszlop megfelelő sorába írom be. Végül, mivel a P2 folyamat az 8. időszeletben érkezett, de csak a 12. időszeletben indult, így 4 időszeletet kellett várnia. Ezt az értéket a Vár oszlop megfelelő helyére írom be.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Érkezik | CPU-idő | Indul | Vége | Vár |
| P1 | 1 | 3 | 2 | 5 | 1 |
| P2 | 3 | 6 | 5 | 8 | 2 |
| P3 | 0 | 2 | 0 | 2 | 0 |
| P4 | 5 | 5 | 8 | 11 | 3 |
| P2 | 8 | 3 | 11 | 14 | 3 |
| P4 | 11 | 2 |  |  |  |

7. lépés: Megkeresem a következő legkorábban érkező folyamatot (P4), majd az indulási idejének megadom az előző folyamat (P2) végzési idejét. Mivel a P4 folyamat CPU-ideje (2) kisebb, mint az időszelet hossza (3), így az indulási időhöz hozzáadom a CPU-időt, majd ezt az értéket (16) a Vége oszlop megfelelő sorába írom be. Végül, mivel a P4 folyamat a 12. időszeletben érkezett, de csak a 15. időszeletben indult, így 3 időszeletet kellett várnia. Ezt az értéket a Vár oszlop megfelelő helyére írom be.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Érkezik | CPU-idő | Indul | Vége | Vár |
| P1 | 1 | 3 | 2 | 5 | 1 |
| P2 | 3 | 6 | 5 | 8 | 2 |
| P3 | 0 | 2 | 0 | 2 | 0 |
| P4 | 5 | 5 | 8 | 11 | 3 |
| P2 | 8 | 3 | 11 | 14 | 3 |
| P4 | 11 | 2 | 14 | 16 | 3 |

7. lépés: Indulási idők növekvő sorendje alapján meghatározható a folyamatok ütemezése:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P3 | 🡪 | P1 | 🡪 | P2 | 🡪 | P4 | 🡪 | P2 | 🡪 | P4 | 🡪 | - | 🡪 | - |

*II/3. Prioritásos*

*Adott öt folyamat P=(1, 2, 3, 4, 5), amelyeknek futásideje C=(3, 6, 2, 1, 3), prioritása pedig Prioritas=(5, 4, 2, 3, 1). A folyamatok T=0 időpontban kerülnek futásra kész állapotba.*

1. lépés: Elkészítem a kiindulási táblázatot a megadott adatok alapján. A folyamatokat az 1. oszlopba, a folyamatok futásra állapotainak idejét (T) a 2. oszlopba, még a futási időt (C) a harmadik oszlopba írom be.

|  |  |  |
| --- | --- | --- |
|  | CPU-idő | Prioritás |
| P1 | 1 | 5 |
| P2 | 2 | 4 |
| P3 | 6 | 2 |
| P4 | 5 | 3 |
| P5 | 3 | 1 |

2. lépés: Mivel minden folyamat a 0. időegységben kerül futásra kész állapotba, így az algoritmusnál a prioritás értéke a mérvadó. Minél kisebb a prioritás annál hamarabb kerül beütemezésre a folyamat. Prioritási sorrendben a folyamatok ütemezése a következő:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P5 | 🡪 | P3 | 🡪 | P4 | 🡪 | P2 | 🡪 | P1 |

*II/4. Legrövidebb hátralévő idejű (Shortest Remaining Time First, SRTF)*

*Adott négy folyamat P=(1, 2, 3, 4), amelyeknek futásideje C=(4, 5, 7, 1). A folyamatok T=(5, 2, 4, 0) időpontban kerülnek futásra kész állapotba. A rendszerben a legrövidebb hátralévő idejű (Shortest Remaining Time First - SRTF) CPU ütemezés működik.*

Adja meg a folyamatok ütemezését:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 🡪 |  | 🡪 |  | 🡪 |  | 🡪 |  | 🡪 |  | 🡪 |  | 🡪 |  |

(Megjegyzés: ha egy döntés után ugyanaz a folyamat folytatódik, akkor azt újból fel kell venni! Ha valamelyik hely már nincs mit ütemezni, akkor jelölje "-" jellel!)

1. lépés: Elkészítem a kiindulási táblázatot a megadott adatok alapján. A folyamatokat az 1. oszlopba, a folyamatok futásra állapotainak idejét (T) a 2. oszlopba, még a futási időt (C) a harmadik oszlopba írom be.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Érkezik | CPU-idő | Indul | Vége | Vár |
| P1 | 5 | 4 |  |  |  |
| P2 | 2 | 5 |  |  |  |
| P3 | 4 | 7 |  |  |  |
| P4 | 0 | 1 |  |  |  |

2. lépés: Megkeresem a legkorábban érkező folyamatot (P4), majd az érkezési idejét, beírom az Indul oszlop megfelelő helyére. Mivel futás közben nem érkezik új folyamat, így a végzési idő egyenlő lesz a P4 folyamat CPU-idejével (1).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Érkezik | CPU-idő | Indul | Vége | Vár |
| P1 | 5 | 4 |  |  |  |
| P2 | 2 | 5 |  |  |  |
| P3 | 4 | 7 |  |  |  |
| P4 | 0 | 1 | 0 | 1 | 0 |

3. lépés: Megkeresem a legkorábban érkező folyamatot (P2), majd az érkezési idejét (2), beírom az Indul oszlop megfelelő helyére, mivel a P4 folyamat már hamarabb befejeződött. Mivel a CPU-idő 5, így egészen a 7. időszeletig tartana a folyamat. Azonban a 4. időszeletben futásra kész állapotba került a P3 folyamat is. Ekkor meg kell vizsgálni, hogy melyik folyamatnak kisebb a szükséges CPU-ideje a 4. időszeletben. Mivel a P2 folyamatnak 3, a P3 folyamatnak pedig 7 a CPU-ideje, így a P2 folyamat folytatódhat tovább. (Ezt jelezni kell a megoldásban úgy, hogy újra leírom a P2 folyamatot.) Az 5. időszeletben azonban futásra kész lesz a P1 folyamat is. Ekkor meg kell vizsgálni, hogy melyik folyamatnak kisebb a szükséges CPU-ideje az 5. időszeletben. Mivel a P2 folyamatnak 2, a P1 folyamatnak pedig 4 a CPU-ideje, így a P2 folyamat folytatódhat tovább. (Ezt jelezni kell a megoldásban úgy, hogy újra leírom a P2 folyamatot.) Ezt követően már nem lesz új folyamat, amely futásra kész áll, így a P2 végzési ideje az érkezési és CPU-idő összege lesz (2+5=7). Végül mivel nem kellett várakoznia, így a 0 értéket kell beírni a Vár oszlop megfelelő oszlopába.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Érkezik | CPU-idő | Indul | Vége | Vár |
| P1 | 5 | 4 |  |  |  |
| P2 | 2 | 5 | 2 | 7 | 0 |
| P3 | 4 | 7 |  |  |  |
| P4 | 0 | 1 | 0 | 1 | 0 |

3. lépés: Mivel a 7. időlépésben vagyok, így meg kell néznem, hogy vannak-e olyan folyamatok, amelyek futásra készen állnak. Ilyen a P1 és a P3 is. Mivel a P1-nek a kisebb a CPU ideje, így őt fogom következőnek beütemezni. P1 indulási ideje meg fog egyezni az előző folyamat (P2) végzési idejével (7). P1 végzési ideje pedig az indulási idő és a CPU-idő összege lesz (7+4=11). Várakozás idő szempontjából az 5. időszeletben kellett volna indulnia, azonban csak a 7. időszeletben tudott, így a várakozási idő 2 lesz.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Érkezik | CPU-idő | Indul | Vége | Vár |
| P1 | 5 | 4 | 7 | 11 | 2 |
| P2 | 2 | 5 | 2 | 7 | 0 |
| P3 | 4 | 7 |  |  |  |
| P4 | 0 | 1 | 0 | 1 | 0 |

4. lépés: Végül az utolsó folyamatot (P3) is beütemezem. P3 indulási ideje meg fog egyezni az előző folyamat (P1) végzési idejével (11). P3 végzési ideje pedig az indulási idő és a CPU-idő összege lesz (11+7=18). Várakozás idő szempontjából az 4. időszeletben kellett volna indulnia, azonban csak a 11. időszeletben tudott, így a várakozási idő 7 lesz.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Érkezik | CPU-idő | Indul | Vége | Vár |
| P1 | 5 | 4 | 7 | 11 | 2 |
| P2 | 2 | 5 | 2 | 7 | 0 |
| P3 | 4 | 7 | 11 | 18 | 7 |
| P4 | 0 | 1 | 0 | 1 | 0 |

5. lépés Folyamatok ütemezésének meghatározása az indulási idők és a folyamatok megállásának és újraindulásának jelölésével.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P4 | 🡪 | P2 | 🡪 | P2 | 🡪 | P2 | 🡪 | P1 | 🡪 | P3 | 🡪 | - | 🡪 | - |

Megjegyzés az algoritmus használatához: Ha egy döntés miatt egy folyamat nem folytatódhat tovább, akkor azt újra fel kell venni a folyamatok közé úgy, hogy a végzési idő lesz az új érkezési idő, a megmaradt CPU-idő pedig az új CPU-idő lesz.