**ZÁRTHELYI 2. ALGORITMUSOK**

**I. Tárkezelés**

*I/1. Legjobban megfelelő (Best Fit, BF)*

Egy rendszerbe korábban már a P1, P2 és P3 folyamatok betöltésre kerültek, amely eredményeként az aktuális memória kiosztása az alábbi:

Cím: 0000 1000 2000 2200 2400 3000 5000 [Byte]

|-------|-------|--------|-------|-------|-------|-------|

Proc: P1 - P3 - P2 - OS

Ha egy 900 Byte méretű P4 és egy 200 Byte méretű P5 folyamat érkezik, akkor a rendszer a best fit algoritmus használatával a P4-t a \_\_\_\_\_\_ címre, a P5-t pedig a \_\_\_\_\_\_ címre tölti be.

Az algoritmus alapja, hogy olyan memóriaterületet válasszunk ki egy folyamat számára, amely a legközelebb áll a folyamat méretéhez, de nem kisebb annál. Az ábrán látható, hogy a P1, P2, P3 folyamatnak korábban már sikerült helyet találnunk, így maradék üres helyekre kell beosztanunk a P4 (900 byte) és P5 (200 byte) folyamatot.

Az üres helyek a következők:

* 1000 és 2000 között
* 2200 és 2400 között
* 3000 és 5000 között

Ezek alapján látható, hogy P4-et az 1000-es címre, míg a P5-öt a 2200-as címre kell elhelyeznünk.

*I/2. Első megfelelő (First Fit, FF)*

Egy rendszerbe korábban már a P1, P2 és P3 folyamatok betöltésre kerültek, amely eredményeként az aktuális memória kiosztása az alábbi:

Cím: 0000 1000 2000 2200 2400 3000 5000 [Byte]

|-------|-------|--------|-------|-------|-------|-------|

Proc: P1 - P3 - P2 - OS

Ha egy 900 Byte méretű P4 és egy 200 Byte méretű P5 folyamat érkezik, akkor a rendszer a first fit algoritmus használatával a P4-et a \_\_\_\_\_\_ címre, a P5-t pedig a \_\_\_\_\_\_ címre tölti be.

Az algoritmus alapja, hogy a memória elejéről kiindulva az első megfelelő méretű memóriaterületet válasszuk ki az adott folyamat számára. Az ábrán látható, hogy a P1, P2, P3 folyamatnak korábban már sikerült helyet találnunk, így maradék üres helyekre kell beosztanunk a P4 (900 byte) és P5 (byte) folyamatot.

Az üres helyek a következők:

* 1000 és 2000 között
* 2200 és 2400 között
* 3000 és 5000 között

Ezek alapján látható, hogy P4-et az 1000-es címre, míg a P5-öt a 2200-as címre elhelyeznünk.

*I/3. Következő megfelelő (Next Fit, NF)*

Egy rendszerbe korábban már a P1, P2 és P3 folyamatok betöltésre kerültek, amely eredményeként az aktuális memória kiosztása az alábbi:

Cím: 0000 1000 2000 2200 2400 3000 5000 [Byte]

|-------|-------|--------|-------|-------|-------|-------|

Proc: P1 - P3 - P2 - OS

Ha egy 900 Byte méretű P4 és egy 200 Byte méretű P5 folyamat érkezik, akkor a rendszer a next fit algoritmus használatával a P4-et a \_\_\_\_\_\_ címre, a P5-öt pedig a \_\_\_\_\_\_ címre tölti be.

Az algoritmus alapja, hogy az első olyan memóriaterületet kell kiválasztanunk az adott folyamat számára, amely elegendő méretű, és a legutóbbi folyamat beillesztése után következik az aktuális pozícióban a memóriában. Ez azt jelenti, hogy az NF algoritmus nem a memóriakezdőponttól, hanem az utolsó beillesztett folyamat után folytatja a keresést. Ha nem talál, akkor kezdi előről a keresést.

Első lépésben a 900 byte méretű P4 folyamatnak kell memóriát foglalnunk. Az utolsó folyamat a P3, amely a 2200-ig foglalja a memóriát, ez a kiindulási alap. 2200 és 2400 között nem fér be. 2400 és 3000 között a memóriát a P2 folyamat foglalja. 3000 és 5000 között van szabad hely, így a P4-et a 3000-es címre helyezem. Következik a P5 folyamat 200 byte mérettel. Mivel a P4 folyamat után már az OS következik, így előről kell kezdem a keresést. 0000 és 1000 között a P1 folyamat szerepel, azonban 1000 és 2000 közé be tudom illeszteni a P5 folyamatot.

Kitekintés: Mikor növekedhet a tördelődés?

* Nem növekszik a külső tördelődés: Ha az új folyamatot közvetlenül az utolsó folyamat után lehet elhelyezni, és nem marad szabad terület az operációs rendszer és az utolsó folyamat között.
* Növekszik a külső tördelődés: Ha az új folyamatot nem lehet közvetlenül az utolsó folyamat után elhelyezni, és az új folyamat beillesztése új szabad területet eredményez az operációs rendszer és az utolsó folyamat között.
* Növekedhet a külső tördelődés: Ha az új folyamat beillesztésekor nem növekszik a külső tördelődés, de az új folyamat beillesztése után más folyamatok elmozdulhatnak a memóriában, ami újabb szabad területeket eredményez az operációs rendszer és az utolsó folyamat között, ezzel növelve a külső tördelődést.

*I/4. Legrosszabban illeszkedő (Worst Fit, WF)*

Egy rendszerbe korábban már a P1, P2 és P3 folyamatok betöltésre kerültek, amely eredményeként az aktuális memória kiosztása az alábbi:

Cím: 0000 1000 2000 2200 2400 3000 5000 [Byte]

|-------|-------|--------|-------|-------|-------|-------|

Proc: P1 - P3 - P2 - OS

Ha egy 900 Byte méretű P4 és egy 200 Byte méretű P5 folyamat érkezik, akkor a rendszer a worst fit algoritmus használatával a P4-et a \_\_\_\_\_\_ címre, a P5-öt pedig a \_\_\_\_\_\_ címre tölti be.

Az algoritmus alapja, hogy a legnagyobb elegendő méretű memóriaterületet válasszuk ki a folyamat számára. Az ábrán látható, hogy a P1, P2, P3 folyamatnak korábban már sikerült helyet találnunk, így maradék üres helyekre kell beosztanunk a P4 (900 byte) és P5 (200 byte) folyamatot.

Az üres helyek a következők:

* 1000 és 2000 között
* 2200 és 2400 között
* 3000 és 5000 között

Ezek alapján látható, hogy P4-et a 3000-es címre, míg a P5-öt az 1000-es címre kell elhelyeznünk.

**II. Virtuális tárkezelés**

*II/1. Legrégebben használt lap (Least Recently Used, LRU)*

A memóriában 1 folyamat fut, amelynek a virtuális memóriája 5 lapból áll. A végrehajtás során a lapokra az alábbi sorrendben hivatkozik: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5. A folyamatok lapkerete 4.

Ha a rendszerben a LRU (legrégebben használt lap) lapcsere stratégiát alkalmazzuk, akkor a folyamat \_\_\_\_\_\_ db laphibával fog lefutni.

Az algoritmus alapja, hogy lapcserekor azt a lapot vesszük ki a memóriából, amelyre a folyamatok leghosszabb ideje nem hivatkoztak.

1. lépés: Kezdeti táblázat felírása a meglévő adatok alapján.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Laphivatkozások | 1 | 2 | 3 | 4 | 1 | 2 | 5 | 1 | 2 | 3 | 4 | 5 |
| Lapok a tárban | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 5 |
|  | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
|  |  | 3 | 3 | 3 | 3 | 5 | 5 | 5 | 5 | 4 | 4 |
|  |  |  | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 3 | 3 |
| Laphibák | x | x | x | x |  |  | x |  |  | x | x | x |

2. lépés: Táblázat kitöltése. Lapcsere az LRU algoritmus alapján:

* Az 1-es bent van a tárban? Nincs. Beírom, és jelzem a laphibát.
* A 2-es bent van a tárban? Nincs. Beírom, és jelzem a laphibát.
* A 3-as bent van a tárban? Nincs. Beírom, és jelzem a laphibát.
* A 4-es bent van a tárban? Nincs. Beírom, és jelzem a laphibát.
* Az 1-es bent van a tárban? Igen. Nincs laphiba.
* A 2-es bent van a tárban? Igen. Nincs laphiba.
* Az 5-ös bent van a tárban? Nincs. Cserélem a legrégebb óta hivatkozott lapra (3), majd jelzem a laphibát.
* Az 1-es bent van a tárban? Igen. Nincs laphiba.
* A 2-es bent van a tárban? Igen. Nincs laphiba.
* A 3-as bent van a tárban? Nincs. Cserélem a legrégebb óta hivatkozott lapra (4), majd jelzem a laphibát.
* A 4-es bent van a tárban? Nincs. Cserélem a legrégebb óta hivatkozott lapra (5), majd jelzem a laphibát.
* Az 5-ös bent van a tárban? Nincs. Cserélem a legrégebb óta hivatkozott lapra (1), majd jelzem a laphibát.

3. lépés: Összeszámolom a laphibák számát, amely összesen 8 db.

*II/2. Második esély (Secound Chance, SC)*

A memóriában 1 folyamat fut, amelynek a virtuális memóriája 5 lapból áll. A végrehajtás során a lapokra az alábbi sorrendben hivatkozik: 1, 2, 3, 4, 3, 1, 5, 1, 2, 3, 1, 5. A folyamatok lapkerete 3. Ha a rendszerben a Second Chance (második esély) lapcsere stratégiát alkalmazzuk, akkor a folyamat...

1. 5 db laphibával fog lefutni.
2. 6 db laphibával fog lefutni.
3. 7 db laphibával fog lefutni.
4. 8 db laphibával fog lefutni.
5. 9 db laphibával fog lefutni.
6. 10 db laphibával fog lefutni.

Az algoritmus alapja, hogy lapcserekor az a lap kerül ki a memóriából, amelyik legrégebb óta bent van, ha még nem hivatkoztak rá. A hivatkozásokat jelölő bittel jelöljük. Ha jelölőbittel ellátott lap a legrégebb óta bent lévő lap, akkor adunk neki egy második esélyt, azonban a jelölő bitjét 0-ra állítjuk.

1. lépés: Kezdeti táblázat felírása a meglévő adatok alapján.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Laphivatkozások | 1 | 2 | 3 | 4 | 3 | 1 | 5 | 1 | 2 | 3 | 1 | 5 |
| Lapok a tárban | 10 | 10 | 10 | 40 | 40 | 40 | 50 | 50 | 50 | 30 | 30 | 30 |
|  | 20 | 20 | 20 | 20 | 10 | 10 | 11 | 11 | 10 | 11 | 10 |
|  |  | 30 | 30 | 31 | 31 | 30 | 30 | 20 | 20 | 20 | 50 |
| Laphibák | x | x | x | x |  | x | x |  | x | x |  | x |

2. lépés: Táblázat kitöltése. Lapcsere az SC algoritmus alapján:

* Az 1-es bent van a tárban? Nincs. Hivatkoztunk már? Nem. Beírom a hivatkozást (1), majd beállítom a jelölőbitet (0), hogy nincs rá hivatkozás. Végül jelzem a laphibát.
* A 2-es bent van a tárban? Nincs. Hivatkoztunk már? Nem. Beírom a hivatkozást (2), majd beállítom a jelölőbitet (0), hogy nincs rá hivatkozás. Végül jelzem a laphibát.
* A 3-es bent van a tárban? Nincs. Hivatkoztunk már? Nem. Beírom a hivatkozást (3), majd beállítom a jelölőbitet (0), hogy nincs rá hivatkozás. Végül jelzem a laphibát.
* A 4-es bent van a tárban? Nincs. Mivel nincs több lapkeret cserélnem kell. Először megnézem, hogy melyik lap van legrégebb óta a tárban. Az 1-es. Mivel nincs hivatkozva (jelölőbit = 0) így, kicserélem rá a 4-est.
* A 3-es bent van a tárban? Igen. Mivel még nem hivatkoztunk rá, de most igen, így a jelölőbitet 1-esre állítom. Nincs laphiba.
* Az 1-es bent van a tárban? Nincs, ezért cserélnem kell. Először megnézem, hogy melyik lap van legrégebb óta a tárban. A 2-es. Mivel nincs hivatkozva (jelölőbit = 0) így, kicserélem rá az 1-est.
* Az 5-ös bent van a tárban? Nincs, ezért cserélnem kell. Először megnézem, hogy melyik lap van legrégebb óta a tárban. A 3-as. Mivel hivatkozva van (jelölőbit = 1) így, kap egy újabb esélyt. A 3-as marad azonban a jelölőbitjét 0-ra állítom. Megnézem, hogy mi a következő legrégebben bent lévő lap. A 4-es. Mivel nincs hivatkozva (jelölőbit = 0) így, kicserélem rá az 5-öst.
* Az 1-es bent van a tárban? Igen. Mivel még nem hivatkoztunk rá, de most igen, így a jelölőbitet 1-esre állítom. Nincs laphiba.
* A 2-es bent van a tárban? Nincs, ezért cserélnem kell. Először megnézem, hogy melyik lap van legrégebb óta a tárban. A 3-as. Mivel nincs hivatkozva (jelölőbit = 0) így, kicserélem rá az 2-est.
* A 3-ös bent van a tárban? Nincs, ezért cserélnem kell. Először megnézem, hogy melyik lap van legrégebb óta a tárban. Az 1-es. Mivel hivatkozva van (jelölőbit = 1) így, kap egy újabb esélyt. Az 1-es marad azonban a jelölőbitjét 0-ra állítom. Megnézem, hogy mi a következő legrégebben bent lévő lap. Az 5-ös. Mivel nincs hivatkozva (jelölőbit = 0) így, kicserélem rá az 3-ast.
* Az 1-es bent van a tárban? Igen. Mivel még nem hivatkoztunk rá, de most igen, így a jelölőbitet 1-esre állítom. Nincs laphiba.
* Az 5-ös bent van a tárban? Nincs, ezért cserélnem kell. Először megnézem, hogy melyik lap van legrégebb óta a tárban. Az 1-es. Mivel hivatkozva van (jelölőbit = 1) így, kap egy újabb esélyt. Az 1-as marad azonban a jelölőbitjét 0-ra állítom. Megnézem, hogy mi a következő legrégebben bent lévő lap. A 2-es. Mivel nincs hivatkozva (jelölőbit = 0) így, kicserélem rá az 5-öst.

3. lépés: Összeszámolom a laphibák számát, amely összesen 9 db.

Megjegyzés: Előfordulhat, hogy az összes lapnak van már jelölőbitje, azonban valamelyik le kel le kell cserélnünk egy új lapra. Ilyenkor mindegyik jelölőbitjét 0-ra állítjuk, majd a legrégebben a tárban lévő lapot cseréljük le az új lapra.

**III. Háttértár**

*III/1. N-lépéses pásztázó (N-Step-SCAN, N-SCAN)*

Egy merevlemezen a cilinderek száma 0-99, a rendszerbe 8 kérés érkezik R= (1, 2, 3, 4, 5, 6, 7, 8), melyek beérkezési időpontja T= (0, 0, 0, 0, 0, 0, 0, 0), és az adatokat tartalmazó cilinderek száma:

C= (33, 27, 8, 47, 82,11, 91, 68). Adja meg a kiszolgálás sorrendjét, ha a háttértár a N-lépéses pásztázó (N-SCAN) algoritmust használja, a fej az egyik cilinderről a másikra 1 ms alatt mozog, és a 0. időpillanatban a 28. cilinder fölött növekvő irányba mozog és N = 3.

A kiszolgálás sorrendje:

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

Az algoritmus alapja, hogy az aktuális cilinderből kiindulva a megadott irányban N számú kérést teljesítünk majd irányt váltunk, ahol újabb N számú kérést teljesítünk. Ha az egyik irányban elfogyott az összes teljesíthető kérés, akkor a másik irány összes kérését teljesítjük.

Az útvonal következő:

* 28 -> 33 (R1)
* 33 -> 47 (R4)
* 47 -> 68 (R8)
* 68 -> 27 (R2)
* 27 -> 11 (R6)
* 11 -> 8 (R3)
* 8 -> 82 (R5)
* 82 -> 91 (R7)

A kiszolgálás sorrendje:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| R1 | 🡪 | R4 | 🡪 | R8 | 🡪 | R2 | 🡪 | R6 | 🡪 | R3 | 🡪 | R5 | 🡪 | R7 |

*III/2. Legrövidebb fejmozgási idő (Shortest seek time first, SSTF)*

Egy merevlemezen a cilinderek száma 0-99, a rendszerbe 8 kérés érkezik R= (1, 2, 3, 4, 5, 6, 7, 8), melyek beérkezési időpontja T= (0, 0, 0, 0, 0, 0, 0, 0), és az adatokat tartalmazó cilinderek száma: C= (33, 27, 8, 47, 82,11, 91, 68). Adja meg a kiszolgálás sorrendjét, ha a háttértár a legrövidebb fejmozgási idő (SSTF) algoritmus használja, a fej az egyik cilinderről a másikra 1 ms alatt mozog, és a 0. időpillanatban a 48. cilinder fölött csökkenő irányba mozog.

A kiszolgálás sorrendje:

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

Az algoritmus alapja, hogy a fej aktuális cilinderből kiindulva arra a cilinderre kell ugranunk, amelyet a legkisebb fejmozgással érhetünk el a mozgási iránynak megfelelően. Ha a fej mozgási irányában már nincsen közelebb cilinder, akkor a fejet irányt vált.

Az útvonal következő:

* 48 -> 47 (R4)
* 47 -> 33 (R1)
* 33 -> 27 (R2)
* 27 -> 11 (R6)
* 11 -> 8 (R3)
* 8 -> 68 (R8)
* 68 -> 82 (R5)
* 82 -> 91 (R7)

A kiszolgálás sorrendje:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| R4 | 🡪 | R1 | 🡪 | R2 | 🡪 | R6 | 🡪 | R3 | 🡪 | R8 | 🡪 | R5 | 🡪 | R7 |