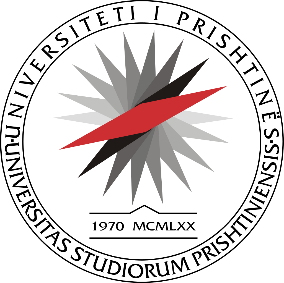
Universiteti i Prishtinës

Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike

Departamenti i Inxhinierisë Kompjuterike



Algoritmi i Bankierit

Begatim Lekaj Prof. Dr. Isak Shabani

Drinor Ahmeti Ass. Artan Mazrekaj

Endrit Hoti

Janar, 2019

Përmbajtja

1.Hyrja………………………………………………………………………………………………………………………………………3

[2. Përshkrimi i Problemit 4](#_Toc534793153)

[3. Implementimi 5](#_Toc534793154)

[4.Bibliografia 15](#_Toc534793156)

1. Hyrja

Një metodë për të zgjidhur problemin e bllokimit(deadlockut) është shmangia e bllokimit.

Kjo realizohet duke kërkuar informata shtesë për mënyren se si do të kërkohen burimet(resurset). Algoritmet të ndryshme shfrytëzojnë këtë çasje..Modeli më i thjeshtë që përdoret kërkon që secili proces të deklaroj numrin maksimal të burimeve të çdo lloji të që mund të ketë nevojë.

Me këtë informacion paraprak është e mundur të ndërtohet një algoritëm që siguron që sistemi asnjëher të mos jetë në një gjendje bllokimi. Ky algoritëm në mënyrë dinamike shqyrton gjendjen e alokimit të resurseve për t'u siguruar që nuk do ndodhë "circular wait" asnjëherë.

Një gjendje është e sigurt nëse sistemi mund të alokon burime të secilit proces (deri në maksimumin e burimeve) në një rend të atillë që të e evitojë bllokimin(deadlockun).

Një sistem është në gjendje të sigurt vetëm nëse ekziston një sekuencë e sigurt. Ndërsa një sekuencë është e sigurt nëse për çdo kërkesë për burime shtesë që bënë një proces mundet ta plotesoj atë kerkesë nga burimet aktualisht në dispozicion dhe nga burimet e mbajtura më heret nga proceset paraprake(përfunduara).

Një gjendje e sigurt nuk është një gjendje e bllokimit(deadlockut).

Një gjendje e bllokuar është gjithmonë një gjendje e pasigurt.

Një gjendje e pasigurt jodomosdoshmërisht është një gjendje e bllokuar por mund të qojë deri tek një gjendje e bllokimit.

Algoritmi i Bankerit është algoritmi më i njohur që shfrytezone këtë çasje për shmangien e bllokimit. Gjatë zbatimit të algoritmit të Bankerit duhet të kemi parasysh 4 struktura të të dhënave:

**Available:**tregon numrin e resurseve të disponueshme(të lira) për secilin lloj të burimeve.

**Max:**tregon numrin maksimal të kërkesave që procesi mund të i bëjë për secilin lloj të burimeve.

**Allocation:**tregon numrin e resurseve të cdo lloji të burimit të alokuar aktualisht për secilin proces.

**Need:**tregon burimet e mbetura të cdo lloji të burimit për secilin process.

2. Përshkrimi i Problemit

Konsideroni pamjen e mëposhtme të sistemit:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Allocation** | | | | **Max** | | | |
| **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |
| **P0** | 2 | 0 | 0 | 1 | 4 | 2 | 1 | 2 |
| **P1** | 3 | 1 | 2 | 1 | 5 | 2 | 5 | 2 |
| **P2** | 2 | 1 | 0 | 3 | 2 | 3 | 1 | 6 |
| **P3** | 1 | 3 | 1 | 2 | 1 | 4 | 2 | 4 |
| **P4** | 1 | 4 | 3 | 2 | 3 | 6 | 6 | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Available** | | | |
| **A** | **B** | **C** | **D** |
| 3 | 3 | 2 | 1 |

Përgjigjuni pyetjeve të mëposhtme duke përdorur algoritmin e bankierit:

1. Tregoni se sistemi është në një gjendje të sigurt duke vertetuar një rend në të cilin proceset mund të kompletohet.
2. Nëse një kërkesë arrin prej procesit P1 për burime shtesë në formën (1,1,0,0), a mund të permbushet kërkesa menjëherë?
3. Nëse një kërkesë arrin prej procesit P4 për burime shtesë në formën (0,0,2,0), a mund të permbushet kërkesa menjëherë?

# 3. Implementimi

Sistemin e ilustruar më poshtë shqyrtojmë nese është në gjendje të sigurtë

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Allocation** | | | | **Max** | | | |
| **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |
| **P0** | 2 | 0 | 0 | 1 | 4 | 2 | 1 | 2 |
| **P1** | 3 | 1 | 2 | 1 | 5 | 2 | 5 | 2 |
| **P2** | 2 | 1 | 0 | 3 | 2 | 3 | 1 | 6 |
| **P3** | 1 | 3 | 1 | 2 | 1 | 4 | 2 | 4 |
| **P4** | 1 | 4 | 3 | 2 | 3 | 6 | 6 | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Available** | | | |
| **A** | **B** | **C** | **D** |
| 3 | 3 | 2 | 1 |

Në fillim fitojmë vleren e matrices **Need** përmes Shprehjes:

Need=Max-Allocation

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Allocation** | | | | **Max** | | | | **Need** | | | |
| **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |
| **P0** | 2 | 0 | 0 | 1 | 4 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| **P1** | 3 | 1 | 2 | 1 | 5 | 2 | 5 | 2 | 2 | 1 | 3 | 1 |
| **P2** | 2 | 1 | 0 | 3 | 2 | 3 | 1 | 6 | 0 | 2 | 1 | 3 |
| **P3** | 1 | 3 | 1 | 2 | 1 | 4 | 2 | 4 | 0 | 1 | 1 | 2 |
| **P4** | 1 | 4 | 3 | 2 | 3 | 6 | 6 | 5 | 2 | 2 | 3 | 3 |

Vektori Available është:

|  |  |  |  |
| --- | --- | --- | --- |
| **Available** | | | |
| **A** | **B** | **C** | **D** |
| 3 | 3 | 2 | 1 |

**Hapi i parë**

Sic shihet nga tabela në vijim plotësohet kushti **Need[0] ≤ Available** ,atëhere rifreskojmë vleren e vektorit Available në këtë mënyrë :. **Available = Available + Allocation[0]** Procesin P0 kompletohet.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Allocation** | | | | **Max** | | | | **Need** | | | |
| **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |
| **P0** | **I Kompletuar ne hapi 1** | | | | | | | | | | | |
| **P1** | 3 | 1 | 2 | 1 | 5 | 2 | 5 | 2 | 2 | 1 | 3 | 1 |
| **P2** | 2 | 1 | 0 | 3 | 2 | 3 | 1 | 6 | 0 | 2 | 1 | 3 |
| **P3** | 1 | 3 | 1 | 2 | 1 | 4 | 2 | 4 | 0 | 1 | 1 | 2 |
| **P4** | 1 | 4 | 3 | 2 | 3 | 6 | 6 | 5 | 2 | 2 | 3 | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Available** | | | |
| **A** | **B** | **C** | **D** |
| 5 | 3 | 2 | 2 |

**Hapi i dytë**

Meqë për proceset P1 dhe P2 nuk plotësohet kushti **Need[i] ≤ Available** , kalojmë në procesin P3 ku nga tabela shihet se plotësohet kushti **Need[i] ≤ Available** , sikurse ne hapin e parë edhe këtu e rifreskojm vektorin Available : **Available = Available + Allocation[3]** Gjithashtu procesi P3 kompletohet .

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Allocation** | | | | **Max** | | | | **Need** | | | |
| **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |
| **P0** | **I Kompletuar ne hapi 1** | | | | | | | | | | | |
| **P1** | 3 | 1 | 2 | 1 | 5 | 2 | 5 | 2 | 2 | 1 | 3 | 1 |
| **P2** | 2 | 1 | 0 | 3 | 2 | 3 | 1 | 6 | 0 | 2 | 1 | 3 |
| **P3** | **I Kompletuar ne hapi 2** | | | | | | | | | | | |
| **P4** | 1 | 4 | 3 | 2 | 3 | 6 | 6 | 5 | 2 | 2 | 3 | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Available** | | | |
| A | B | C | D |
| 6 | 6 | 3 | 4 |

**Hapi i tretë**

Pasi sic shihet nga tabela kushti **Need[4] ≤ Available** plotësohet ,atëhere rifreskojmë vleren e vektorit Available : **Available = Available + Allocation[4]** Procesin P4 kompletohet.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Allocation** | | | | **Max** | | | | **Need** | | | |
| **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |
| **P0** | **I Kompletuar në hapi 1** | | | | | | | | | | | |
| **P1** | 3 | 1 | 2 | 1 | 5 | 2 | 5 | 2 | 2 | 1 | 3 | 1 |
| **P2** | 2 | 1 | 0 | 3 | 2 | 3 | 1 | 6 | 0 | 2 | 1 | 3 |
| **P3** | **I Kompletuar në hapi 2** | | | | | | | | | | | |
| **P4** | **I Kompletuar ne hapi 3** | | | | | | | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Available** | | | |
| **A** | **B** | **C** | **D** |
| 7 | 10 | 6 | 6 |

**Hapi i katërt**

Tani ktheme te procesi P1 dhe shqyrtojm nese tash mund ta plotëson kushtin **Need[1] ≤ Available** , pas rifreskimit të shumfisht të vektorit Available . Nga tabela shihet se kushti **Need[1] ≤ Available** plotësohet, atëhere rifreskojmë përsëri vleren e vektorit Available : **Available = Available + Allocation[4]** Gjithashtu përfundojmë punën me procesin P1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Allocation** | | | | **Max** | | | | **Need** | | | |
| **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |
| **P0** | **I Kompletuar në hapi 1** | | | | | | | | | | | |
| **P1** | **I Kompletuar në hapi 4** | | | | | | | | | | | |
| **P2** | 2 | 1 | 0 | 3 | 2 | 3 | 1 | 6 | 0 | 2 | 1 | 3 |
| **P3** | **I Kompletuar në hapi 2** | | | | | | | | | | | |
| **P4** | **I Kompletuar në hapi 3** | | | | | | | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Available** | | | |
| **A** | **B** | **C** | **D** |
| 10 | 11 | 8 | 7 |

**Hapi i pestë**

Sikurse te procesi P1 rikthehemi edhe te procesi P2 dhe shqyrtojm nese tash mund ta plotëson kushtin **Need[2] ≤ Available** , pas rifreskimit të shumfisht të vektorit Available . Nga tabela shihet se plotesohet kushti **Need[2] ≤ Available** , atëhere rifreskojmë përsëri vleren e vektorit Available : **Available = Available + Allocation[2]** Gjithashtu procesin P2 kompletohet.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Allocation** | | | | **Max** | | | | **Need** | | | |
| **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |
| **P0** | **I Kompletuar ne hapi 1** | | | | | | | | | | | |
| **P1** | **I Kompletuar ne hapi 4** | | | | | | | | | | | |
| **P2** | **I Kompletuar ne hapi 5** | | | | | | | | | | | |
| **P3** | **I Kompletuar ne hapi 2** | | | | | | | | | | | |
| **P4** | **I Kompletuar ne hapi 3** | | | | | | | | | | | |

Vlera perfundimtare e vektorit Available është:

|  |  |  |  |
| --- | --- | --- | --- |
| **Available** | | | |
| **A** | **B** | **C** | **D** |
| 12 | 12 | 8 | 10 |

Sekuenca e sigurt është :

P0🡪P3🡪P4🡪P1🡪P2

Pra, sistemi eshte ne gjendje te sigurte pra nuk ka bllokim(deadlock).

**b.)**

Kërkesa që vjen nga procesi P1 për burime shtesë P1 🡪 (1,1,0,0), pra për dy instance shtesë ,një për tipin e burimit(resursit) A dhe një të tipit të burimit B është e ilustruar në tabelen në vijim:

|  |  |  |  |
| --- | --- | --- | --- |
| **Request** | | | |
| **A** | **B** | **C** | **D** |
| 1 | 1 | 0 | 0 |

Në ndërkohë gjendja aktuale e sistemit dhe e matricës Need është si vijon:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Allocation** | | | | **Max** | | | | **Need** | | | |
| **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |
| **P0** | 2 | 0 | 0 | 1 | 4 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| **P1** | 3 | 1 | 2 | 1 | 5 | 2 | 5 | 2 | 2 | 1 | 3 | 1 |
| **P2** | 2 | 1 | 0 | 3 | 2 | 3 | 1 | 6 | 0 | 2 | 1 | 3 |
| **P3** | 1 | 3 | 1 | 2 | 1 | 4 | 2 | 4 | 0 | 1 | 1 | 2 |
| **P4** | 1 | 4 | 3 | 2 | 3 | 6 | 6 | 5 | 2 | 2 | 3 | 3 |

Vektori Available që përcakton numrin e instancave të cdo burimi në një kohë të caktuar mbanë vlerat:

|  |  |  |  |
| --- | --- | --- | --- |
| **Available** | | | |
| **A** | **B** | **C** | **D** |
| 3 | 3 | 2 | 1 |

Duke përdorur algoritmin për përcaktimin nëse kërkesa mund të përmbushet shqyrtojm rastet:

1. Nëse **Request[i] ≤Need[i]** ,kalojmë në hapin e dytë
2. Nëse **Reques[i] ≤Available** ,kalojmë në hapin e tretë
3. Modifikojm gjendjen si në vazhdim:

**Available = Available–Request[i];**

**Allocation[i] = Allocation[i] + Request[i];**

**Need[i] = Need[i] –Request[i];**

Në rastin tonë : Need[1] =(2,1,3,1) . Available = (3,3,2,1) .Në rastin tonë përmbushen të dy kushtet e para , pasi kërkesa P1🡪 (1,1,0,0) është me e vogël ose e barabartë me vleren e Need[1]= (2,1,3,1) dhe vleren e Available = (3,3,2,1).

Pas zbatimit të kërkesës për resurce shtesë nga ana e procesit P1, tabela e sistemit pas rifreskimit duket si më poshtë:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Allocation** | | | | **Max** | | | | **Need** | | | |
| **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |
| **P0** | 2 | 0 | 0 | 1 | 4 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| **P1** | **4** | **2** | **2** | **1** | 5 | 2 | 5 | 2 | **1** | **0** | **3** | **1** |
| **P2** | 2 | 1 | 0 | 3 | 2 | 3 | 1 | 6 | 0 | 2 | 1 | 3 |
| **P3** | 1 | 3 | 1 | 2 | 1 | 4 | 2 | 4 | 0 | 1 | 1 | 2 |
| **P4** | 1 | 4 | 3 | 2 | 3 | 6 | 6 | 5 | 2 | 2 | 3 | 3 |

Vektori Available :

|  |  |  |  |
| --- | --- | --- | --- |
| **Available** | | | |
| **A** | **B** | **C** | **D** |
| **2** | **2** | **2** | **1** |

Tani përmes algoritmit të siguris sikurse ne detyren nën **a.)** shqytojmë nese gjendja e re është e sigurt(duke gjetur sekuencën e sigurt) .

**Hapi i parë**

Pasi sic shihet nga tabela vlera **Need[0] ≤ Available** , atëhere rifreskojmë vleren e vektorit Available në këtë mënyrë :. **Available = Available + Allocation[0]** Gjithashtu procesi P0 kompletohet.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Allocation** | | | | **Max** | | | | **Need** | | | |
| **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |
| **P0** | **I Kompletuar ne hapi 1** | | | | | | | | | | | |
| **P1** | 3 | 1 | 2 | 1 | 5 | 2 | 5 | 2 | 2 | 1 | 3 | 1 |
| **P2** | 2 | 1 | 0 | 3 | 2 | 3 | 1 | 6 | 0 | 2 | 1 | 3 |
| **P3** | 1 | 3 | 1 | 2 | 1 | 4 | 2 | 4 | 0 | 1 | 1 | 2 |
| **P4** | 1 | 4 | 3 | 2 | 3 | 6 | 6 | 5 | 2 | 2 | 3 | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Available** | | | |
| **A** | **B** | **C** | **D** |
| **4** | **2** | **2** | **2** |

**Hapi i dytë**

Meqë për proceset P1 dhe P2 nuk plotësohet kushti **Need[i] ≤ Available** , kalojmë në procesin P3 ku tabela sihet se plotësohet kushti **Need[3] ≤ Available** ,sikurse ne hapin e parë edhe këtu e rifreskojm vektorin Available : **Available = Available + Allocation[3]** Gjithashtu procesi P3 kompletohet .

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Allocation** | | | | **Max** | | | | **Need** | | | |
| **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |
| **P0** | **I Kompletuar ne hapi 1** | | | | | | | | | | | |
| **P1** | 3 | 1 | 2 | 1 | 5 | 2 | 5 | 2 | 2 | 1 | 3 | 1 |
| **P2** | 2 | 1 | 0 | 3 | 2 | 3 | 1 | 6 | 0 | 2 | 1 | 3 |
| **P3** | **I Kompletuar ne hapi 2** | | | | | | | | | | | |
| **P4** | 1 | 4 | 3 | 2 | 3 | 6 | 6 | 5 | 2 | 2 | 3 | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Available** | | | |
| **A** | **B** | **C** | **D** |
| **5** | **5** | **3** | **4** |

**Hapi i tretë**

Pasi sic shihet nga tabela kushti **Need[4] ≤ Available** plotësohet ,atëhere rifreskojmë vleren e vektorit Available : **Available = Available + Allocation[4]** Gjithashtu procesi P4 kompletohet.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Allocation** | | | | **Max** | | | | **Need** | | | |
| **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |
| **P0** | **I Kompletuar ne hapi 1** | | | | | | | | | | | |
| **P1** | 3 | 1 | 2 | 1 | 5 | 2 | 5 | 2 | 2 | 1 | 3 | 1 |
| **P2** | 2 | 1 | 0 | 3 | 2 | 3 | 1 | 6 | 0 | 2 | 1 | 3 |
| **P3** | **I Kompletuar ne hapi 2** | | | | | | | | | | | |
| **P4** | **I Kompletuar ne hapi 3** | | | | | | | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Available** | | | |
| **A** | **B** | **C** | **D** |
| **6** | **9** | **6** | **6** |

**Hapi i katërt**

Tani kthehemi te procesi P1 dhe shqyrtojm nese tash mund ta plotëson kushtin **Need[1] ≤ Available** , pas rifreskimit të shumfisht të vektorit Available . Nga tabela shihet se kushti **Need[1] ≤ Available** plotësohet, atëhere rifreskojmë përsëri vleren e vektorit Available : **Available = Available + Allocation[4]** Gjithashtu procesi P1 kompletohet.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Allocation** | | | | **Max** | | | | **Need** | | | |
| **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |
| **P0** | **I Kompletuar ne hapi 1** | | | | | | | | | | | |
| **P1** | **I Kompletuar ne hapi 4** | | | | | | | | | | | |
| **P2** | 2 | 1 | 0 | 3 | 2 | 3 | 1 | 6 | 0 | 2 | 1 | 3 |
| **P3** | **I Kompletuar ne hapi 2** | | | | | | | | | | | |
| **P4** | **I Kompletuar ne hapi 3** | | | | | | | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Available** | | | |
| **A** | **B** | **C** | **D** |
| **10** | **11** | **8** | **7** |

**Hapi i pestë**

Sikurse te procesi P1 rikthehemi edhe te procesi P2 dhe shqyrtojm nese tash mund ta plotëson kushtin **Need[2] ≤ Available** , pas rifreskimit të shumfisht të vektorit Available . Nga tabela shihet se plotesohet kushti **Need[2] ≤ Available** , atëhere rifreskojmë përsëri vleren e vektorit Available : **Available = Available + Allocation[2]** Gjithashtu procesi P2 kompletohet.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Allocation** | | | | **Max** | | | | **Need** | | | |
| **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |
| P0 | **I Kompletuar ne hapi 1** | | | | | | | | | | | |
| P1 | **I Kompletuar ne hapi 4** | | | | | | | | | | | |
| P2 | **I Kompletuar ne hapi 5** | | | | | | | | | | | |
| P3 | **I Kompletuar ne hapi 2** | | | | | | | | | | | |
| P4 | **I Kompletuar ne hapi 3** | | | | | | | | | | | |

Vlera perfundimtare e vektorit Available është:

|  |  |  |  |
| --- | --- | --- | --- |
| **Available** | | | |
| **A** | **B** | **C** | **D** |
| **12** | **12** | **8** | **10** |

Pra, kerkesa P1🡪(1,1,0,0) mund të përmbushet menjëherë , meqenëse edhe sistemi është në gjendje të sigurtë sepse ekziston sekuenca e sigurtë.

Në rastin tone sekuenca e sigurtë është : P0🡪P3🡪P4🡪P1🡪P2.

Sekuencat tjera të sigurta janë P0🡪P3🡪(të gjitha kombinimet që mund të bëhen me proceset P3,P4,P5).

**c)**

Kërkesa që vjen nga procesi P4 për burime shtesë P4 🡪 (0,0,2,0), pra për dy instance shtesë për tipin e burimit(resursit) C është e ilustruar në tabelen në vijim:

|  |  |  |  |
| --- | --- | --- | --- |
| **Kerkesa** | | | |
| **A** | **B** | **C** | **D** |
| 0 | 0 | 2 | 0 |

Pamja e sistemit është:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Allocation** | | | | **Max** | | | | **Need** | | | |
| **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |
| **P0** | 2 | 0 | 0 | 1 | 4 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| **P1** | 3 | 1 | 2 | 1 | 5 | 2 | 5 | 2 | 2 | 1 | 3 | 1 |
| **P2** | 2 | 1 | 0 | 3 | 2 | 3 | 1 | 6 | 0 | 2 | 1 | 3 |
| **P3** | 1 | 3 | 1 | 2 | 1 | 4 | 2 | 4 | 0 | 1 | 1 | 2 |
| **P4** | 1 | 4 | 3 | 2 | 3 | 6 | 6 | 5 | 2 | 2 | 3 | 3 |

Vlera e vektorit available është:

|  |  |  |  |
| --- | --- | --- | --- |
| **Available** | | | |
| **A** | **B** | **C** | **D** |
| 3 | 3 | 2 | 1 |

Duke përdorur algoritmin për përcaktimin nëse kërkesa mund të përmbushet shqyrtojm rastet:

1. Nëse **Request[i] ≤Need[i]** ,kalojmë në hapin e dytë
2. Nëse **Reques[i] ≤Available** ,kalojmë në hapin e tretë
3. Modifikojm gjendjen si në vazhdim:

**Available = Available–Request[i];**

**Allocation[i] = Allocation[i] + Request[i];**

**Need[i] = Need[i] –Request[i];**

Në rastin tonë : Need[1] =(0,0,2,0) . Available = (3,3,2,1)

Në rastin tonë përmbushen të dy kushtet e para , pasi kërkesa P4🡪 (0,0,2,0) është me e vogël ose e barabartë me vleren e Need[1]= (2,1,3,1) dhe vleren e Available = (3,3,2,1).

Pas zbatimit të kërkesës për resurce shtesë nga ana e procesit P1, tabela e sistemit pas rifreskimit duket si më poshtë:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Allocation** | | | | **Max** | | | | **Need** | | | |
| **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C…...** | **D** |
| **P0** | 2 | 0 | 0 | 1 | 4 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| **P1** | 4 | 2 | 2 | 1 | 5 | 2 | 5 | 2 | 1 | 0 | 3 | 1 |
| **P2** | 2 | 1 | 0 | 3 | 2 | 3 | 1 | 6 | 0 | 2 | 1 | 3 |
| **P3** | 1 | 3 | 1 | 2 | 1 | 4 | 2 | 4 | 0 | 1 | 1 | 2 |
| **P4** | 1 | 4 | 5 | 2 | 3 | 6 | 6 | 5 | 2 | 2 | 1 | 3 |

Vlera e vektori Available është:

|  |  |  |  |
| --- | --- | --- | --- |
| **Available** | | | |
| **A** | **B** | **C** | **D** |
| 3 | 3 | 0 | 1 |

Tani përmes algoritmit të siguris shqytojmë nese gjendja e re është e sigurt(duke gjetur sekuencën e sigurt) .

Meqenëse për asnjë nga proceset nuk përmbushet kushti : **Need[i] ≤ Available** atëhere nuk ekziston një sekuencë e sigurtë, e për rrjedhim sistemi është I pasigurtë. Kështu që kërkesa e procesit P4 për burime shtesë nuk mund të përmbushet.

# 4. Bibliografia

Silberschatz A. (2018). *Operating System Concepts* (10th ed.). Hoboken: Peter Baer Galvin & Greg Gagne.

Chauhan N. (2014). *Principles of Operating Systems*. Oxford: Oxford University Press.