Starea de interblocare

Evitarea Interblocarii

Detectarea interblocarii

Metode mixte

Conceptul de interblocare

- Resursele logice (fişiere, baze de date, semafoare etc.) sau fizice (imprimante, spaţiul de memorie internă, memorii externe, cicluri UC etc.) pot fi formate din unul sau mai multe elemente. Etapele parcurse de un proces în cursul utilizării unei resurse sunt:
  - **cerere de acces**: dacă cererea nu poate fi satisfăcută imediat, procesul care a formulat-o va fi nevoit să aştepte până când poate dobândi resursa;
  - **utilizare**: procesul poate folosi resursa;
  - **eliberare**: procesul eliberează resursa.
- Cererea şi eliberarea de resurse se face prin **apeluri de sistem**. Evidenţa alocării resurselor se realizează prin intermediul unei **tabele de sistem.** Dacă un proces cere o resursă ale cărei elemente sunt alocate altor procese din sistem, atunci procesul care a efectuat cererea va fi pus într-o coadă de aşteptare, asociată resursei respective.
- Se spune că un set de procese se află în stare de **interblocare** atunci când orice proces din setul respectiv se află în aşteptarea unui eveniment de eliberare a unei resurse cerute, ce poate fi produs numai de către un proces aflat în mulţimea respectivă.
- Pentru rezolvarea problemei interblocării se folosesc, în principiu două metode.
- **- prevenirea** interblocării: constă în utilizarea unui protocol care să nu permită niciodată sistemului să intre în starea de interblocare.
- - **detectarea** interblocări: permite sistemului intrarea în starea de interblocare şi apoi rezolvă această problemă.

Condiţii necesare pentru apariţia interblocării

- excludere mutuală: există cel puţin o resursă ocupată în mod exclusiv, adică fără a putea fi folosită în comun de către un singur proces; dacă un alt proces formulează o cerere pentru aceeaşi resursă, va fi nevoit să aştepte până în momentul eliberării ei.

- ocupare şi aşteptare: există cel puţin un proces care ţine ocupată cel puţin o resursă şi aşteaptă să obţină resurse suplimentare ocupate în acel moment de către alte procese.

- imposibilitatea achiziţionării forţate: resursele nu pot fi achiziţionate forţat de către un proces de la un alt proces care le ocupă în acel moment; resursele pot fi eliberate numai de către procesele care le ocupă, decât după ce acestea şi-au îndeplinit sarcinile.

- aşteptare circulară: în sistem există un set de procese aflate în starea de aşteptare, (p_1,p_2,..,p_n), astfel încât p_1 aşteaptă eliberarea unei resurse ocupate de către p_2, p_2 aşteaptă eliberarea unei resurse ocupate de către p_3, ş.a.m.d. p_{n-1} aşteaptă eliberarea unei resurse ocupate de către p_n, p_n aşteaptă eliberarea unei resurse ocupate de către p_1.

- Se observă că ultima condiţie implică şi cerinţa de ocupare şi aşteptare, astfel încât cele patru condiţii nu sunt complet independente; cu toate acestea este util ca fiecare condiţie să fie discutată şi tratată separat.

Evitarea interblocării

- **Starea alocării resurselor** este definită de nr. de resurse disponibile şi alocate şi de nr. maxim de cereri de resurse formulate de către procese.
- Se spune că o stare este **sigură** dacă sistemul poate aloca fiecărui proces resursele cerute (până la numărul maxim), într-o anumită ordine şi evitând apariţia interblocării.
- Sistemul se află într-o stare sigură numai dacă există o **secvenţă sigură**. Se spune că o secvenţă de procese $(p_1, p_2, .., p_n)$ este o secvenţă sigură pentru starea de alocare curentă, dacă pentru fiecare p_i, resursele pe care acesta le-ar mai putea cere pot fi alocate dintre resursele disponibile, la care se mai adaugă resursele deţinute de către toate celelalte procese pj, cu j<i. În acest caz, dacă resursele cerute de către procesul p_i nu sunt disponibile imediat, acesta va trebui să aştepte până când toate procesele p_j, cu j<i îşi încheie execuţia. În acest moment p_i poate obţine toate resursele de care are nevoie, îşi termină sarcina, eliberează resursele şi se încheie, după care procesul p_{i+1} poate obţine resursele pe care le doreşte ş.a.m.d.
- În cazul în care nu există o astfel de secvenţă, se spune că starea sistemului este nesigură.
- O stare de interblocare este o stare nesigură; nu toate stările nesigure sunt interblocări, dar o stare nesigură poate conduce la interblocare.
- Într-o stare nesigură, sistemul de operare nu poate împiedica procesele să formuleze în aşa fel cererile de alocare a resurselor încât acestea să ducă la interblocare.

- **Exemplu.**
- Să considerăm un sistem în care 4 procese p_1, p_2, p_3, p_4 folosesc în comun o resursă cu 20 de elemente. Pe durata întregii execuţii procesele au nevoie de maximum 14, 6, 10 şi respectiv 9 elemente. Iniţial, procesele formulează o cerere pentru 7, 3, 4 şi respectiv 2 elemente. Analizând situaţia sistemului la momentul iniţial se constată că el se află într-o stare sigură, deoarece, de exemplu, secvenţa p_2, p_1, p_3, p_4 este o secvenţă sigură.
- **Justificare**. După alocarea iniţială, mai rămăn 20-16=4 elemente. Pentru a fi executat p_2, există suficiente elemente ale resursei (6<4+3), adică elementele resursei deţinute de proces după alocarea iniţială, la care se adaugă cele disponibile. După ce procesul p_2 îşi termină execuţia, sunt disponibile 7 elemente ale resursei, care adăugate la cele 7 alocate iniţial lui p_1 fac posibilă execuţia acestuia ş.a.m.d.
- Dacă procesul p_4 cere încă 2 elemente şi acestea îi sunt alocate, se trece nu numai într-o stare nesigură dar şi într-o stare de interblocare, deoarece numărul de resurse libere, egal cu 2, este insuficient pentru oricare dintre procesele aflate în aşteptare.
- Algoritmul de evitare a interblocării pentru cazul resurselor **cu mai multe elemente**, care va fi descris în continuare, se numeşte **algoritmul bancherului**.

Algoritmul bancherului

- Pentru implementarea algoritmului sunt necesare câteva structuri de date care să codifice starea de alocare a resurselor sistemului. Dacă n este numărul de procese din sistem şi m este numărul de resurse, se definesc:
  - D(„Disponibil”): vector de dimensiune m care indică numărul de elemente disponibile ale resurselor: D(j) conţine numărul elementelor disponibile ale resursei r_j, j=1,..,m;
  - M(„Maxim”): matrice de dimensiune nxm care indică numărul maxim de cereri care pot fi formulate de către fiecare proces; M(i,j) conţine numărul maxim de elemente ale resursei r_j cerute de procesul p_i;
  - A(„Alocare”): matrice de dimensiune nxm care indică numărul de elemente din fiecare resursă care sunt alocate în mod curent fiecărui proces; A(i,j) conţine numărul de elemente ale resursei r_j alocate procesului p_i;
  - N(„Necesar”): matrice de dimensiune nxm care indică numărul de elemente ce ar mai putea fi necesare fiecărui proces; N(i,j) conţine numărul de elemente ale resursei r_j de care ar mai avea nevoie procesul p_i pentru a-şi realiza sarcina; evident că:

$$N(i,j) = M(i,j) - A(i,j).$$

- Fie C_i vectorul cererilor formulate de către procesul p_i; $C_i(j)$ este numărul cererilor formulate de procesul p_i din resursa r_j; în momentul în care procesul p_i formulează o cerere de resurse, vor fi parcurse urm. etape:

  Pas 1. Dacă $C_i \leq N_i$, se execută pasul 2, altfel, se consideră că a apărut o eroare, deoarece procesul a depăşit cererea maxim admisibilă.

  Pas 2. Dacă $C_i \leq D$, se execută pasul 3, altfel, p_i este nevoit să aştepte (resursele nu sunt disponibile).

  Pas 3. Se simulează alocarea resurselor cerute de procesul p_i şi starea se modifică astfel:

  $$D:=D-C_i;\ Ai:=A_i+C_i;\ N_i:=N_i-C_i.$$

- Pentru verificarea **stării de siguranţă** a sistemului se foloseşte algoritmul următor. Algoritmul foloseşte:

  - vectorul de lucru L, de dimensiune m ale cărui componente vor corespunde resurselor sistemului; componenta L(i) va conţine numărul resurselor r_i disponibile la un moment dat.

  - un vector T de dimensiune n, care corespunde proceselor şi care va marca procesele parcurse.

  - un vector s de dimensiune n; componenta s(i) va conţine indicele (poziţia) din secvenţă a procesului.

Pas 1(Iniţializări). `L:=D`; T(i):=0, pentru `i=1,..,`n; citeşte `s`.

Pas 2. (Parcurgerea secvenţei de procese**)**

```
K:=0;
   do { k:=k+1;
   if (N(s(k)) <=L) then
       { L:=L+A( s(k));
         T(s(k)):=1        }
   until k=n or not T(s(k))
       }
```

Pas 3. (Verificare dacă toate proc. din secv. au fost marcate cu `true`). Dacă `T(i):=1` pentru `i=1,..,`n, atunci sist. se află într-o stare sigură.

- **Observatii.**

  1. Algoritmul constă în parcurgerea secvenţei de procese, simularea terminării execuţiei unui proces, adică adăugarea la disponibilul de resurse existent înaintea terminării procesului de la pasul respectiv a resurselor procesului care tocmai şi-a terminat execuţia.

  2. Dacă starea de alocare a resurselor rezultată este sigură, se alocă procesului p_i resursele cerute. În caz contrar, procesul p_i este nevoit să aştepte, iar sistemul reface starea de alocare a resurselor existentă înainte de execuţia **pasului 3**.

Exemplu

- Să presupunem că în sistem avem 5 procese p_1, p_2, p_3, p_4, p_5 şi patru resurse r_1, r_2, r_3, r_4, care au 3, 14, 12 respectiv 13 elemente. Se consideră că starea iniţială este definită de:

$$A = \begin{pmatrix} 0 & 0 & 1 & 2 \\ 1 & 0 & 0 & 0 \\ 1 & 3 & 5 & 4 \\ 0 & 6 & 3 & 2 \\ 0 & 0 & 1 & 4 \end{pmatrix} \qquad M = \begin{pmatrix} 0 & 0 & 1 & 2 \\ 1 & 7 & 5 & 0 \\ 2 & 3 & 5 & 6 \\ 0 & 6 & 5 & 2 \\ 0 & 6 & 5 & 6 \end{pmatrix}$$

$$N = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 7 & 5 & 0 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & 2 & 0 \\ 0 & 6 & 4 & 2 \end{pmatrix}$$

$$D = \begin{pmatrix} 1 & 5 & 2 & 0 \end{pmatrix}$$

- **Iteraţia 1**(procesul p_1): $(0\ \ 0\ \ 0\ \ 0) \le (1\ \ 5\ \ 2\ \ 0)\ T = (1\ \ 0\ \ 0\ \ 0\ \ 0)$

$$L = (0\ \ 0\ \ 1\ \ 2) + (1\ \ 5\ \ 2\ \ 0) = (1\ \ 5\ \ 3\ \ 2)$$

- **Iteraţia 2**(procesul p_3): $(1\ \ 0\ \ 0\ \ 2) \le (1\ \ 5\ \ 3\ \ 2)\ T = (1\ \ 0\ \ 1\ \ 0\ \ 0)$

$$L = (1\ \ 3\ \ 5\ \ 4) + (1\ \ 5\ \ 3\ \ 2) = (2\ \ 8\ \ 8\ \ 6)$$

- **Iteraţia 3**(procesul p_4): $(0\ \ 0\ \ 2\ \ 0) \le (2\ \ 8\ \ 8\ \ 6)\ T = (1\ \ 0\ \ 1\ \ 1\ \ 0)$

$$L = (0\ \ 6\ \ 3\ \ 2) + (2\ \ 8\ \ 8\ \ 6) = (2\ \ 14\ \ 11\ \ 8)$$

- **Iteraţia 4**(procesul p_5): $(0\ \ 6\ \ 4\ \ 2) \le (2\ \ 14\ \ 11\ \ 8)\ T = (1\ \ 0\ \ 1\ \ 1\ \ 1)$

$$L = (0\ \ 0\ \ 1\ \ 4) + (2\ \ 14\ \ 11\ \ 8) = (2\ \ 14\ \ 12\ \ 12)$$

- **Iteraţia 5**(procesul p_2): $(0\ \ 7\ \ 5\ \ 0) \le (2\ \ 14\ \ 12\ \ 12)\ T = (1\ \ 1\ \ 1\ \ 1\ \ 1)$

$$L = (1\ \ 0\ \ 0\ \ 0) + (2\ \ 14\ \ 12\ \ 12) = (3\ \ 14\ \ 12\ \ 12)$$

- **Pasul 3** $T = (1\ \ 1\ \ 1\ \ 1\ \ 1)$ deci sistemul se află într-o stare sigură

- În cazul în care procesul p_2 formulează o cerere suplimentară pentru 4, repectiv 2 elemente din r_2 , respectiv r_3, trebuie să se verifice dacă această cerere poate fi satisfăcută. Conform algoritmului bancherului, se verifică îndeplinirea relaţiilor:

$$C_2 \leq N_2, \text{ adică } (0,4,2,0) \leq (0,7,5,0)$$

$$C_2 \leq D, \text{ adică } (0,4,2,0) \leq (1,5,2,0)$$

- Cum inegalităţile sunt îndeplinite, se simulează alocarea. Starea sistemului devine:

$$A = \begin{pmatrix} 0 & 0 & 1 & 2 \\ 1 & 4 & 2 & 0 \\ 1 & 3 & 5 & 4 \\ 0 & 6 & 3 & 2 \\ 0 & 0 & 1 & 4 \end{pmatrix} \qquad N = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 3 & 3 & 0 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & 2 & 0 \\ 0 & 6 & 4 & 2 \end{pmatrix}$$

$$D = \begin{pmatrix} 1 & 1 & 0 & 0 \end{pmatrix}$$

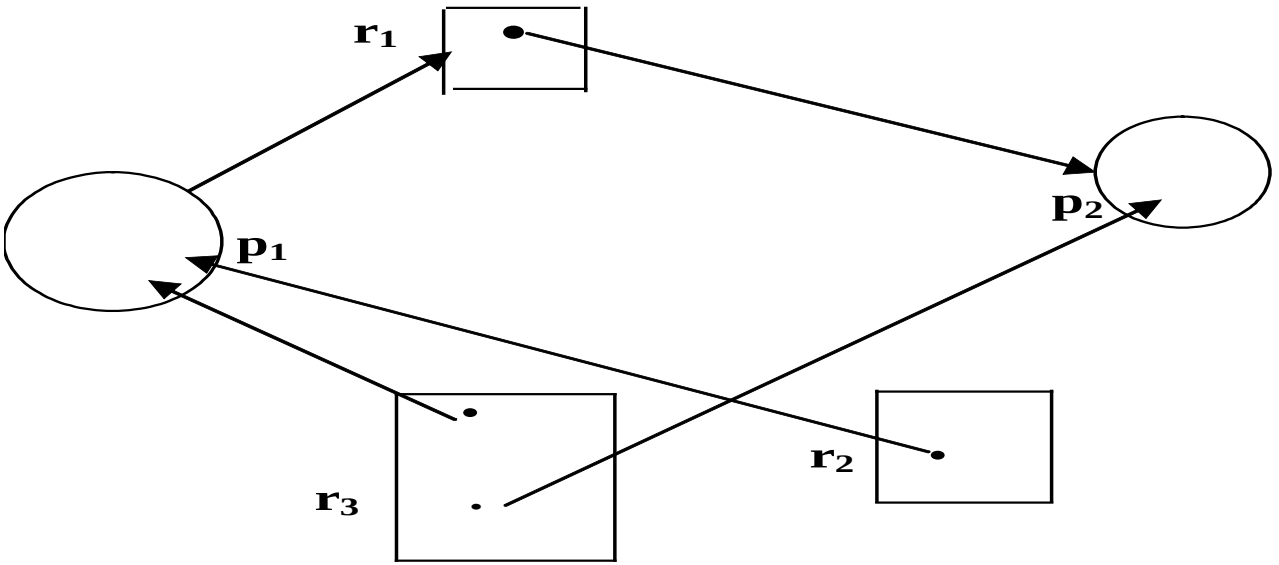
- Temă. Să se arate că această stare a sistemului este sigură.

- Indicaţie. Se consideră secvenţa $(p_1, p_3, p_4, p_2, p_5)$.

Graful de alocare a resurselor

- Algoritmul bancherului are o complexitate de ordinul $m \times n^2$; pentru evitarea interblocării, în cazul resurselor cu un singur element se utilizează un algoritm de complexitate mult mai mică, care va fi descris în continuare.
- **Graful de alocare a resurselor** este de forma $G=(N,A)$, în care $N=P \cup R$, $P=\{p_1,..,p_n\}$ fiind mulţimea proceselor iar $R=\{r_1,..,r_m\}$ mulţimea resurselor. Un arc poate fi de forma (p_i,r_j) sau (r_j,p_i); arcul (p_i,r_j), numit **arc cerere,** are semnificaţia că procesul p_i a cerut un element al resursei r_j, iar arcul (r_j,p_i), numit **arc alocare,** înseamnă că un element al resursei r_j a fost alocat procesului p_i.
- În cadrul grafului, procesele sunt reprezentate grafic prin cercuri, iar resursele prin pătrate sau dreptunghiuri.
- Deoarece resursele pot fi formate din mai multe elemente, fiecare element se reprezintă cu ajutorul unui punct plasat în interiorul pătratului sau dreptunghiului respectiv.
- Atunci când procesul p_i formulează o cerere pentru un element al resursei r_j, se inserează în graful alocare a resurselor un arc cerere. În momentul în care cererea este satisfăcută, arcul cerere este transformat în arc alocare. Atunci când procesul eliberează resursa, arcul alocare este şters din graf.

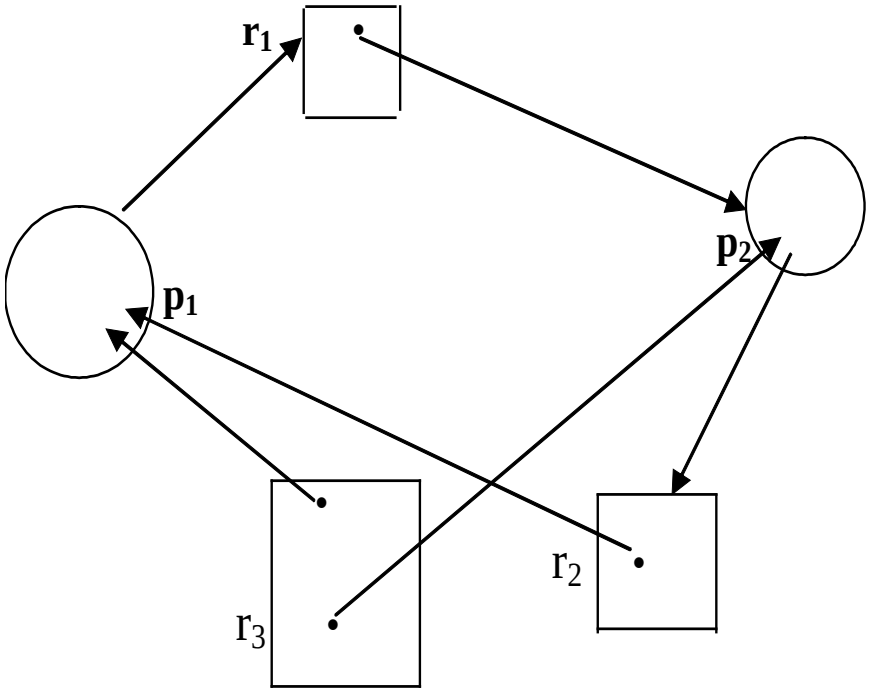
- **Exemplu.** Considerăm un sistem cu două resurse r_1 şi r_2, care au câte un element şi o resursă r_3 cu două elemente, în care există două procese, p_1 şi p_2. Avem deci: $P=\{p_1, p_2\}$, $R=\{r_1, r_2, r_3\}$. Dacă $A=\{(p_1, r_1), (r_1, p_2), (r_2, p_1), (r_3, p_1), (r_3, p_2)\}$, reprezentarea grafică a grafului este cea din figura urmatoare



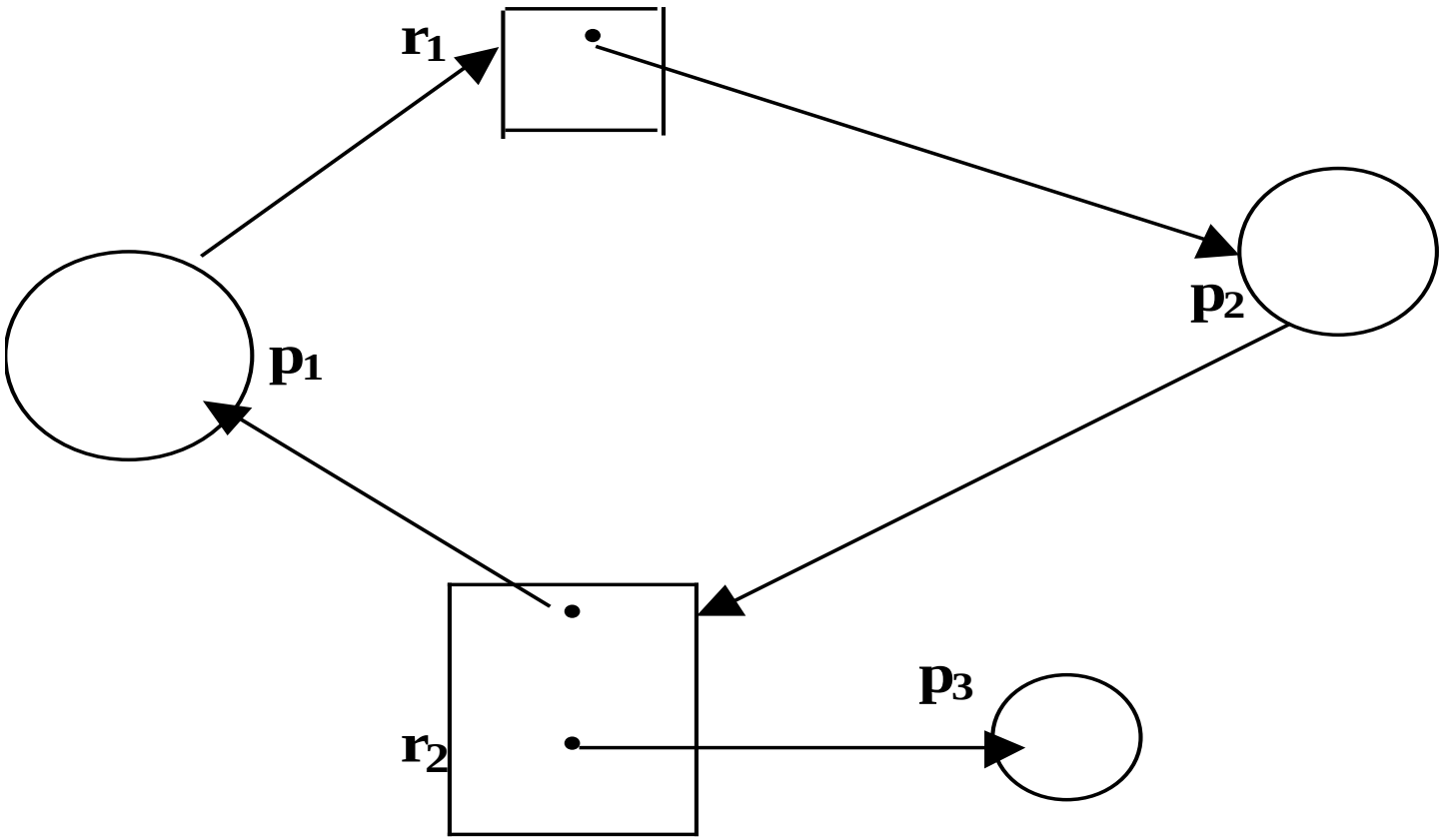
- Mulţimea de arce ilustrează starea proceselor:
- procesul p_1 are alocată câte un element al resurselor r_2 şi r_3 şi aşteaptă obţinerea unui elem. al resursei r_1;
- procesul p_2 are alocată resursa r_1 şi un element al resursei r_3.

- Cunoscând definiţia grafului de alocare a resurselor, se poate arăta că:
  - Dacă graful nu conţine circuite, în sistem nu există interblocare.
  - Dacă există un circuit, atunci poate să apară interblocarea.
  - Dacă există resurse cu mai multe elemente, atunci existenţa unui circuit nu implică în mod necesar apariţia interblocării, aceasta fiind doar o condiţie necesară dar nu şi suficientă.

  Exemplu. Vom folosi din nou situaţia prezentată anterior, dar se presupune în plus că că procesul p_2 formulează o cerere suplimentară(figura urmatoare). Deoarece resursa de tip r_2 cerută de procesul p_2 nu este disponibilă, fiind deja alocată procesului p_1, în graf se inserează arcul cerere (p_2, r_2), formându-se circuitul $(p_1, r_1, p_2, r_2, p_1)$. În acest moment, cele două procese intră în starea de interblocare.



- În figura următoare, este prezentată o situaţie în care, deşi există un circuit $(p_1, r_1, p_2, r_2, p_1)$, nu există interblocare. După ce p_3 îşi termină execuţia, p_2 poate primi r_2 ,şi după terminarea execuţiei lui cedează r_1 care poate fi folosită de către p_1 şi care poate astfel să-şi termine execuţia.
-



Detectarea interblocării

- Atunci când sistemul nu utililizează algoritmi de evitare a interblocării, deorece utilizarea lor ar mări costurile, se pot utiliza metodele de detectare şi revenire, care însă, pe lângă cheltuielile datorate utilizării unor structuri de date suplimentare, presupun şi cheltuieli datorate revenirii din starea de interblocare.
- Problema care se pune, este când să fie apelat algoritmul de detectare a interblocării. Dacă acest lucru se face ori de câte ori se formulează o cerere de resurse, efectul este creşterea substanţială a timpului de calcul.
- O variantă mai puţin costisitoare este apelarea algoritmului la anumite **intervale de timp** sau atunci când **gradul de utilizare a CPU** scade sub un anumit prag (de exemplu 40 la sută), justificarea fiind că o interblocare poate conduce la „paralizarea” funcţionării sistemului şi deci la o scădere a gradului de utilizare a CPU.

Algoritmul de detectare pentru resurse cu mai multe elemente

- Algoritmul foloseşte structurile de date D, A şi C cu aceeşi semnificaţie ca şi în cazul algoritmului de evitare a interblocării.
- Algoritmul prezentat ia în consideraţie toate procesele a căror execuţie nu s-a încheiat încă şi analizează toate secvenţele posibile de alocare a resurselor pentru aceste procese.

  Pas 1. Fie L şi T vectori de dimensiune m, respectiv n. Pentru i=1,..,n:

  L(i):=D(i);

  Dacă $A_i \neq 0$, atunci T(i):=false, altfel T(i):=true.

  Pas 2. Se caută i astfel încât: T(i)=false şi $C_i \leq L$. Dacă nu există, **goto pas 4**.

  Pas 3. Se execută secvenţa: L:=L+A_i; T(i):=true; **goto pas 2**.

  Pas 4. Dacă există i (i=1,..,n) a.i. T(i)=false, sistemul se află în starea de interblocare. În plus procesul pentru care T(i)=false, provoacă interblocarea.
- **Obs.** La **pasul 3** sunt preluate resursele deţinute de procesul p_i, în cazul în care la **pasul 2** se îndeplineşte condiţia $C_i \leq L$, adica p_i nu este implicat în mod curent într-o interblocare; deci, se presupune că p_i nu va mai cere alte resurse pentru a-şi termina execuţia şi va elibera toate resursele care i-au fost alocate. Altfel, la următoarea utilizare a algoritmului se poate sa fie depistată o interblocare.

- **Exemplu** Presupunem că avem trei procese p_1, p_2, p_3, care folosesc în comun trei resurse r_1, r_2, r_3, având câte 4, 2 şi respectiv 2 elemente.
- De asemenea, valorile structurilor de date, care dau starea iniţială sunt:

$$A=\begin{pmatrix}2 & 2 & 1\\ 1 & 0 & 1\\ 1 & 0 & 0\end{pmatrix} \qquad C=\begin{pmatrix}0 & 0 & 1\\ 0 & 0 & 0\\ 2 & 1 & 0\end{pmatrix} \qquad D=\begin{pmatrix}0 & 0 & 0\end{pmatrix}$$

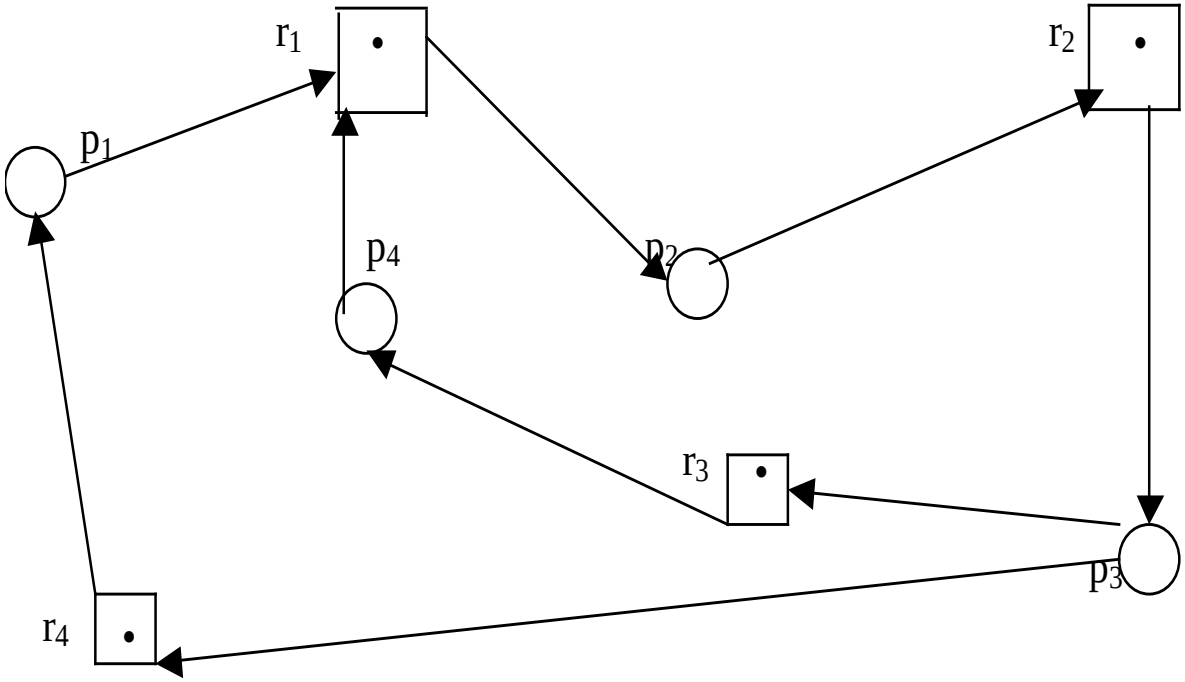
- Să verificăm că nu există interblocare; vom aplica algoritmul prezentat pe secvenţa (p_2, p_1, p_3)
- **Pas 1**. `T=(0,0,0); L=(0,0,0).`
- **Paşii 2 şi 3.**
- **Iteraţia 1.** Pentru procesul p_2 avem $\begin{pmatrix}0 & 0 & 0\end{pmatrix}\le\begin{pmatrix}0 & 0 & 0\end{pmatrix}$, deci:
  $L:=\begin{pmatrix}1 & 0 & 1\end{pmatrix}$ $T:=\begin{pmatrix}0 & 1 & 0\end{pmatrix}$
- **Iteraţia 2.** Pentru procesul p_1 avem, $\begin{pmatrix}0 & 0 & 1\end{pmatrix}\le\begin{pmatrix}1 & 0 & 1\end{pmatrix}$, deci :
  $L:=\begin{pmatrix}1 & 0 & 1\end{pmatrix}+\begin{pmatrix}2 & 2 & 1\end{pmatrix}=\begin{pmatrix}3 & 2 & 3\end{pmatrix}$ $T:=\begin{pmatrix}1 & 1 & 0\end{pmatrix}$
- **Iteraţia 3.** Pentru procesul p_3 avem, $\begin{pmatrix}1 & 0 & 0\end{pmatrix}\le\begin{pmatrix}3 & 2 & 3\end{pmatrix}$, deci ,
  $L:=\begin{pmatrix}1 & 0 & 0\end{pmatrix}+\begin{pmatrix}3 & 2 & 3\end{pmatrix}=\begin{pmatrix}4 & 2 & 3\end{pmatrix}$ $T:=\begin{pmatrix}1 & 1 & 1\end{pmatrix}$
- **Pas 4.** $T:=\begin{pmatrix}1 & 1 & 1\end{pmatrix}$, deci nu există interblocare în sistem.

- Dacă însă procesul p_1 formulează o cerere suplimentară pentru 2 elemente ale lui r_1, modificând prima linie a matricii `C` , care va conţine acum valorile `2,0,1`, sistemul intră în starea de interblocare.
- Chiar dacă procesul p_1 îşi încheie execuţia şi eliberează resursele care i-au fost alocate `(1,0,1),` numărul total de resurse disponibile nu este suficient pentru a putea acoperi necesarul formulat de oricare dintre celelalte două procese. Prin urmare, procesele p_1 şi p_3 sunt interblocate.

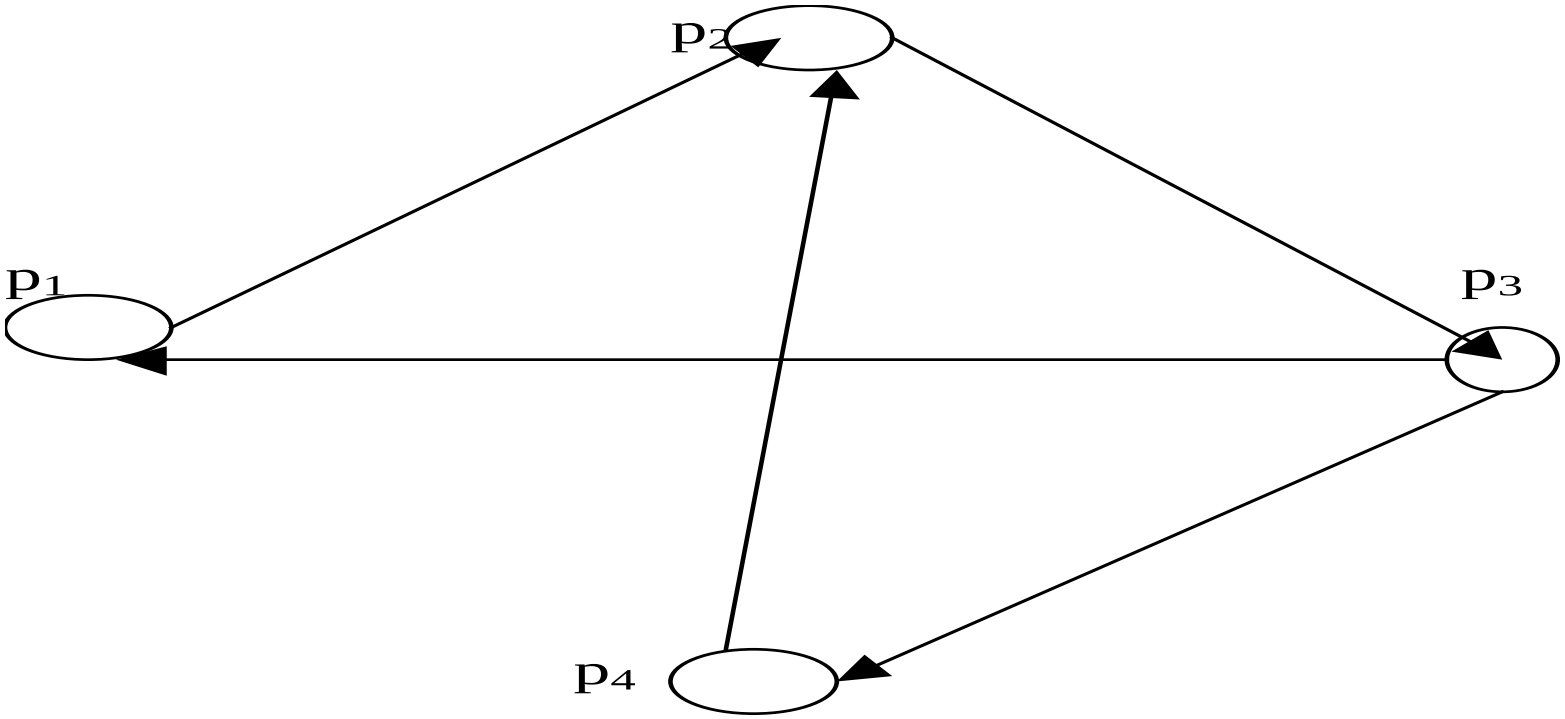
Algoritmul de detectare a interblocării pentru resurse cu un singur element

- Algoritmul de detectare a interblocării pentru resurse cu mai multe elemente necesită un număr de operaţii de ordinul mxn^2.
- Dacă toate resursele au câte un singur element, se poate defini un algoritm care utilizează o variantă a grafului de alocare a resurselor, numit **graful ”aşteaptă-pentru”.**
- Acesta se obţine din graful de alocare a resurselor prin eliminarea nodurilor resursă şi contopirea arcelor corespunzătoare: un arc (p_i, p_j) indică faptul că p_i aşteaptă ca p_j să elibereze resursa care îi este necesară. Într-un graf ”aşteaptă-pentru”, un arc (p_i, p_j) există dacă şi numai dacă, pentru o resursă r_k oarecare, există două arce în graful alocării resurselor asociat (p_i, r_k) şi (r_k, p_j).

- **Exemplu.** În figura urmatoare este prezentat un graf de alocare a resurselor



- **Graful ”aşteaptă-pentru” asociat.** Folosind această metodă, se poate afirma că în sistem există interblocare dacă şi numai dacă graful „aşteaptă-pentru” conţine un circuit.
- Pentru a detecta interblocările, sistemul trebuie să actualizeze graful „aşteaptă-pentru” şi să apeleze periodic un algoritm care să testeze dacă există circuite în acest graf.
- Acest algoritm necesită un număr de n^2 operaţii, deci complexitatea este mai mică decât a algoritmului general.
- Totuşi, există un dezavantaj, şi anume operaţiile suplimentare necesare reprezentării şi actualizării grafului „aşteaptă-pentru”.



Ieşirea din interblocare

- **I. Terminarea Proceselor.**
- 1. Sunt abandonate toate procesele implicate în interblocare.
- 2. Se abandonează câte un proces dintre cele care au provocat interblocarea, până când această stare este eliminată.
- **Criterii de abandonare a proceselor**.

  a. Prioritatea proceselor.

  b. Timpul scurs de la lansarea în exec. a procesului şi timpul rămas până la terminarea acestuia.
- c. Resorsele utilizate de proces.
- d. Resorsele necesare procesului ptr. a fi terminat.
- e. Dacă procesul este interactiv sau nu.

- **II. Achiziţionarea forţată a resurselor.**
- Această metodă asigură eliminarea stării de interblocare prin achiziţionarea succesivă a resurselor utilizate de anumite procese şi alocarea lor altor procese, până când se elimină interblocarea.
- Principalele probleme care trebuie rezolvate în acest caz sunt:

  - alegerea proceselor cărora li se vor lua resursele, după criterii care să conducă la costuri minime(numărul resurselor ocupate, timpul de execuţie consumat);

  - reluarea execuţiei: un proces căruia i s-au luat forţat anumite resurse nu-şi mai poate continua normal execuţia, ea trebuie reluată dintr-un moment anterior, când a primit prima dintre resursele luate;

  - evitarea „înfometării”, adică să nu fie selectat pentru achiziţonare forţată a resurselor un acelaşi proces.

Metode mixte de tratare a interblcărilor

- Practica a dovedit că nici una dintre metodele de bază prezentate anterior, nu poate acoperi toate cauzele care produc interblocarea.
- O modalitate de rezolvare a acestei probleme, este combinarea algoritmilor de bază prezentaţi.
- Metoda propusă are la bază ideea că resursele pot fi grupate în clase şi în fiecare clasă se utilizează metoda de gestionare a resurselor cea mai adecvată.
- Datorită grupării resurselor în clase, o interblocare nu poate implica mai mult decât o clasă de resurse
- În interiorul unei clase se aplică una dintre metodele de bază.
- Dacă interiorul oricărei clase nu apare interblocare, în sistem nu pot să apară interblocări.

- **Exemplu.**Se poate considera un sistem format din patru clase de resurse:

  - resurse interne, adică resursele utilizate de către sistem (de exemplu, PCB-urile proceselor);

  - memoria internă;

  - resursele job-ului: de exemplu, drivere de dispozitiv (disc) şi fişiere;

  - spaţiul din memoria auxiliară alocat fiecărui job utilizator.
- O metodă mixtă pentru rezolvarea interblocării în cazul acestui sistem, ordonează clasele descrise anterior, folosind în cadrul fiecărei clase următoarele abordări:

  - prevenirea interblocării prin achiziţionare forţată a spaţiului din memoria internă (se poate evacua oricând un job în memoria auxiliară);

  - evitarea interblocării în cazul resurselor job-ului (informaţiile necesare despre formularea cererilor de resurse pot fi obţinute din liniile de comandă);

  - alocarea prealabilă a spaţiului din memoria auxiliară asociat fiecărui job utilizator(se cunoaşte necesarul maxim de memorie externă al fiecărui job).