Rappel : stratégies utilisées face à l'interblocage:

- **Méthode d'évitement (Algorithme du Banquier) :**

Un état de système est définie par :

- `Rmax[1..m]` : Nombre total, initial, de ressources.
- `Annonce[p,*]` : Nombre maximal de ressources nécessaires au processus p.
- `Allocate[p,*]` : Nombre de ressources couramment allouées au processus p.
- `Available[1..m]` : Nombre de ressources disponibles.

Algorithme Banquier
Debut
(1) Chercher P_i non marqué tel que : **Need[i,*]=(Annonce[i,*] – Allocate[i,*]) ≤ Aavailable**.
(2) Si P_i n'existe pas:
(3) Aller en 7.
(4) **Available ← Available + Allocate[i,*]**.
(5) Marquer P_i.
(6) Aller en 1.
(7) Si il reste un processus non marqué:
(8) Il est en situation d'interblocage.
Fin.

- **Méthode de détection (Algorithme TestSain) :**

Un état de système est définie par :

- `Rmax[1..m]` : Nombre total, initial, de ressources.
- `Request[p,*]` : Nombre de ressources demandées et non encore obtenues par le processus p.
- `Allocate[p,*]` : Nombre de ressources couramment allouées au processus p.
- `Available[1..m]` : Nombre de ressources disponibles.

Algorithme TestSain
Debut
(1) Chercher P_i non marqué tel que : **Request[i,*] ≤ Available**.
(2) si P_i n'existe pas:
(3) aller en 7.
(4) **Available ← Available + Allocate[I,*]**.
(5) Marquer P_i.
(6) Aller en 1.
(7) si il reste un processus non marqué:
(8) il est en situation d'interblocage.
Fin.

Remarque :

Si tous les processus sont marqués (sélectionnée) après l'exécution de l'algorithme de **Banquier** alors
- Il n'existe pas un risque d'inteblocage.
- On dit que l'état du système est fiable.
- L'ordre d'exécution des processus, appelé suite fiable, est défini par l'ordre de marquage des processus

Sinon, il y a un risque d'interbloquage et on dit que l'état n'est pas fiable.
Les processus non marqués sont les processus qui sont en interblocages.

Si tous les processus sont marqués marqués après l'exécution de l'algorithme **TestSain** alors :
- Il n'existe pas un risque d'inteblocage.
- On dit que l'état du système est sain
- L'ordre d'exécution des processus, appelé suite saine, est défini par l'ordre de marquage des processus

Sinon, il y a un risque d'interbloquage et on dit que l'état n'est pas fiable. Les processus non marqués sont les processus qui sont en interblocages.

Exercice 01 : Soit un système utilisant la **détection** d'interblocages À l'instant **t1** l'état du système est présenté par les matrices suivants:

	Allocate			Request		
P_0	0	1	0	0	0	0
P_1	2	0	0	2	0	2
P_2	3	0	3	0	0	0
P_3	2	1	1	1	0	0
P_4	0	0	2	0	0	2

Available		
R_1	R_2	R_3
0	0	0

1. Calculer Rmax.
2. Déterminer une suite saine complète pour que l'état du système soit sain.
3. Supposons que P2 fait la demande request p2 = [0 0 1] :
   - Calculer la nouvelle valeur de vecteur Available**.**
   - Le système reste-t-il toujours en état sain? Si oui donner la suite saine sinon donner les processus qui sont en interblocages.

Réponse :

1- Rmax = Available+∑Allocate(Pi,*)= Available+ Allocate(P_0,*)+ Allocate(P_1,*)+ Allocate(P_2,*)+ Allocate(P_3,*)+ Allocate(P_4,*)+ Allocate(P_5,*)= (0,0,0)+(0,1,0)+(2,0,0)+(3,0,3)+(2,1,1)+(0,0,2)=(7,2,6);

2- Algorithme **TestSain** pour déterminer une suite saine (Au départ tous les processus sont non marqué , à chaque étape on va marquer (sélectionner) le premier processus Pi vérifiant l'équation request ($P_{i,*}$)<= available)

Etape 01 : request($P_{0,}$*) =(0,0,0)<= Available **=(0,0,0) alors**

Available= Available+Allocate($P_{0,}$*) = (0,0,0)+(0,1,0)=(0,1,0) ;

Marquer (barrer) $P_{0.}$ La suite saine : S={ P_0};

Etape 02 : request($P_{2,}$*) =(0,0,0)<= Available **=(0,1,0); alors**

Available= Available+Allocate($P_{2,}$*)=(0,1,0)+(3,0,3)=(3,1,3).

Marquer (barrer) P_2 et ajouter P_2 dans la suite saine : S={ $P_{0,}$ P_2};

Etape 03 : request($P_{1,}$*) =(2,0,2)<= Available **=(3,1,3); alors**

Available= Available+Allocate($P_{1,}$*)=(3,1,3)+(2,0,0)=(5,1,3).

Marquer (barrer) P_1 → S={ $P_{0,}$ $P_{2,}$ P_1};

Etape 04 : request($P_{3,}$*) =(1,0,0)<= Available **=(5,1,3); alors**

Available= Available+Allocate($P_{3,}$*)=(5,1,3)+(2,1,1)=(7,2,4).

Marquer (barrer) P_3 → S={ $P_{0,}$ $P_{2,}$ $P_{1,}$ P_3 };

Etape 05 : request($P_{4,}$*) =(0,0,2)<= Available **=(7,2,4); alors**

Available= Available+Allocate($P_{3,}$*)=(7,2,4)+(0,0,2)=(7,2,6)=Rmax

Marquer (barrer) P_4 → S={ $P_{0,}$ $P_{2,}$ $P_{1,}$ $P_{3,}$ P_4 };

La Suite Saine est : S={ $P_{0,}$ $P_{2,}$ $P_{1,}$ $P_{3,}$ P_4 }

3- P2 fait la demande request p2 = [0 0 1], et on a Rmax = Available+∑Allocate(Pi,*) → Available = Rmax -∑Allocate(Pi,*) alors pas de relation entre Request et Available ainsi Available ne changera pas→ Available =(0,0,0).

Pour vérifier si le système reste-t-il toujours en état sain, on applique l'algorithme Testsain.

Ainsi en "Etape 01" on a request($P_{0,}$*) =(0,0,0)<= Available =(0,0,0) et

Available= Available+Allocate(P0,*) = (0,0,0)+(0,1,0)=(0,1,0) → Available = (0,1,0).

En "Etape 02" in n'exista pas un processus P_i qui vérifier request ($P_{i,*}$)<= available=(0,1,0) ainsi l'algorithme Testsain s'arrête à la 2ème étape et les processus P_1 , P_2 ,P_3 et P_4 sont en interbloquage.

Exercice 02 : Soit un système utilisant la méthode d'évitement . À un instant donné, le système est dans l'état suivant :

	Allocate				Annonce			
P_0	1	0	0	0	1	1	1	0
P_1	0	1	0	0	0	1	1	0
P_2	0	0	0	1	1	1	1	1

Rmax			
1	1	1	1

1. Vérifiez en utilisant l'algorithme du banquier, si l'état courant est fiable.

Réponse :

Calcul de la matrice Need :

Need = Annonce **-** Allocate **; on soustrait les éléments de la matrice** Annonce des éléments correspondants de la matrice Allocate. Voici le résultat

	Need			
P_0	0	1	1	0
P_1	0	0	1	0
P_2	1	1	1	0

4- Algorithme du Banquier pour déterminer une suite saine

Available = Rmax -∑Allocate(Pi,*) = (1,1,1,1)-(1,1,0,1)=**(0,0,1,0)**

Etape 01 : Need(P_1 , *)=(0,0,1,0)<= Available ; alors

Available= Available+Allocate($P_{1,}$*)=(0,0,1,0)+(0,1,0,0)=**(0,1,1,0)**.

Marquer (barrer) P_1 → La suite faible: F={ P_1};

Etape 02 : Need(P_0 , *)=(0,1,1,0)<= Available=(0,1,1,0) ; alors

Available= Available+Allocate($P_{0,}$*)=(0,1,1,0)+(1,0,0,0)=**(1,1,1,0)**.

Marquer (barrer) P_0 → La suite faible: F={ $P_{1,}$ P_0};

Etape 03 : Need(P_2 , *)=10,1,1,0)<= Available=(1,1,1,0) ; alors

Available= Available+Allocate($P_{2,}$*)=(1,1,1,0)+(0,0,0,1)=**(1,1,1,1)**=Rmax

Marquer (barrer) P_2 → La suite faible: F={ $P_{1,}$ $P_{0,}$ P_2};

La Suite faible est : F={ $P_{1,}$ $P_{0,}$ P_2}