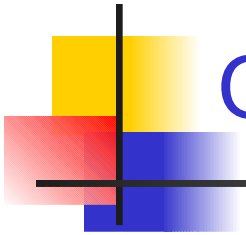


Luc Courtrai

CONCURRENCE

Université de Bretagne SUD
UFR SSI - Departement MIS

Concurrence

Plan

Notion de processus

Les appels système UNIX

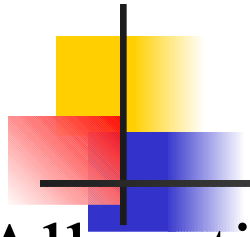
La synchronisation entre processus

La communication entre processus

Problème d'interblocage

Les threads JAVA

Les Posix threads



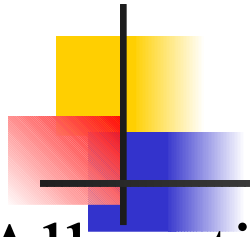
Concurrence : Interblocage

Allocation des ressources

Les processus utilisent et se partagent un ensemble de ressources (cpu, mémoire,segment, page, disque, entrée, fichier, périphériques …

l'allocation :

- **statique (début du processus)**
- **dynamique (demande et libération)**



Concurrence : Interblocage

Allocation des ressources

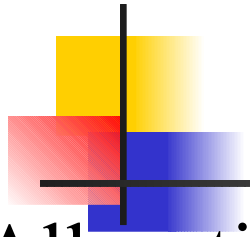
Quotas disques (statique)

- quotas de 1GO pour 1000 utilisateurs (1To) sur un disque de 500Go

- Allocation effective à la création des fichiers

Allocation mémoire (dynamique)

- le malloc (brk) alloue le mémoire mais les pages ne sont rééllement affectées qu'au premier accès en écriture.



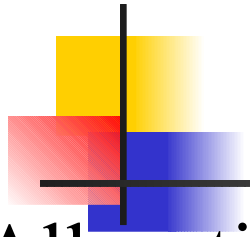
Concurrence : Interblocage

Allocation des ressources

Interblocage

=

Un ensemble de processus est en interblocage lorsque chacun attend une ressource déjà allouée pour un processus de l'ensemble



Concurrence : Interblocage

Allocation des ressources

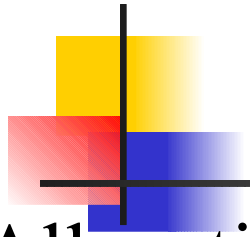
Exemple

3 processus effectuent des copies de bandes magnétiques et le système dispose de 3 lecteurs de bandes.

interblocage : chaque processus a réservé un des lecteurs et reste bloqué sur la réservation du second

2 processus P1 et P2 demande deux ressources de types différents R1 R2 dans l'ordre inverse

interblocage : P1 obtient R1 et P2 R2



Concurrence : Interblocage

Allocation des ressources

Interblocage = phénomène stable qui persiste indéfiniment

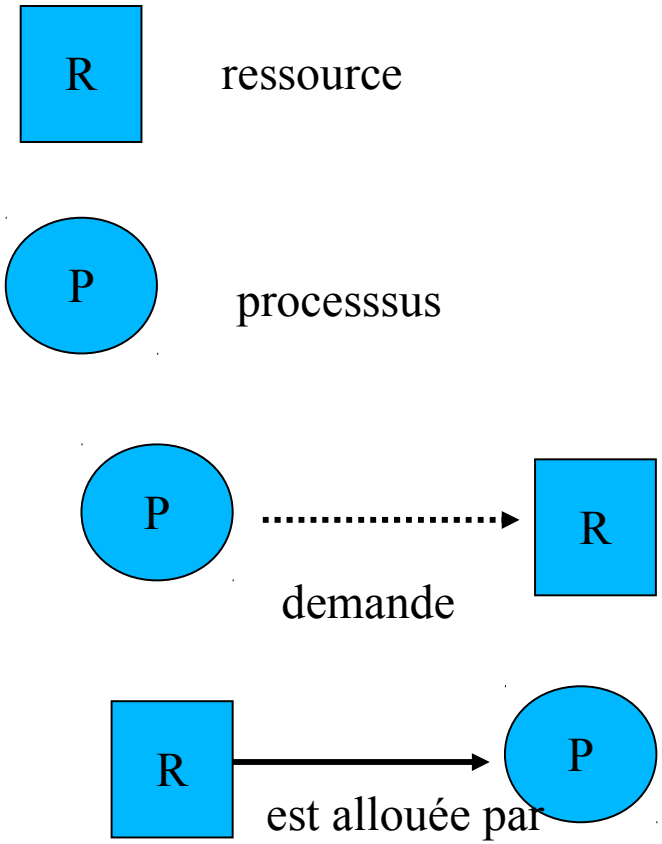
Le phénomène gagne les autres processus qui vont demander une des ressources bloquées

L'OS doit détecter, puis traiter l'interblocage

détruire un ou plusieurs processus pour libérer une ou plusieurs ressources et dont débloquer l'interblocage.

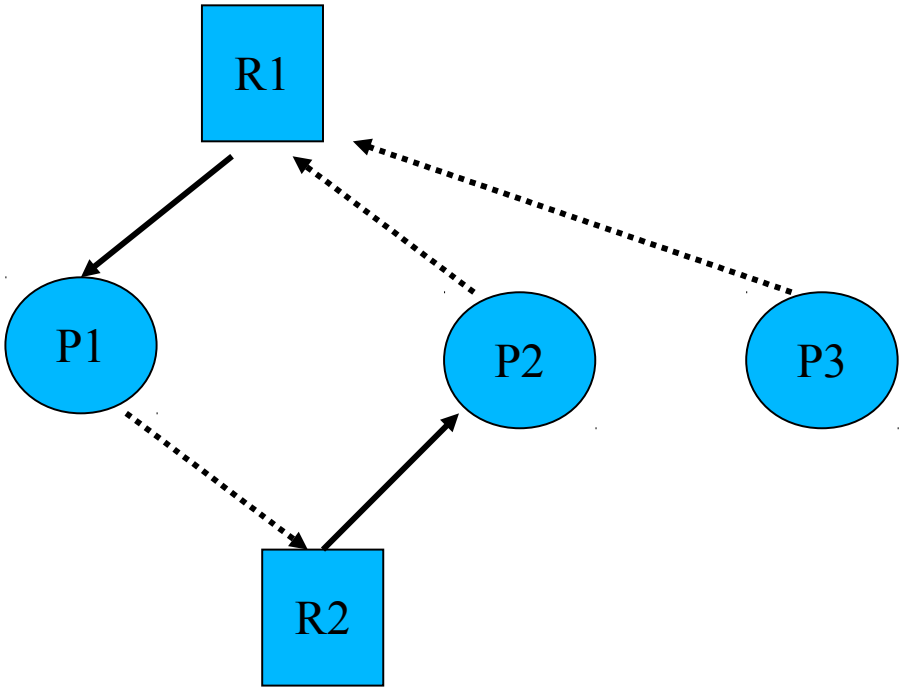
Concurrence : Interblocage

graphe des allocation/demande



Concurrence : Interblocage

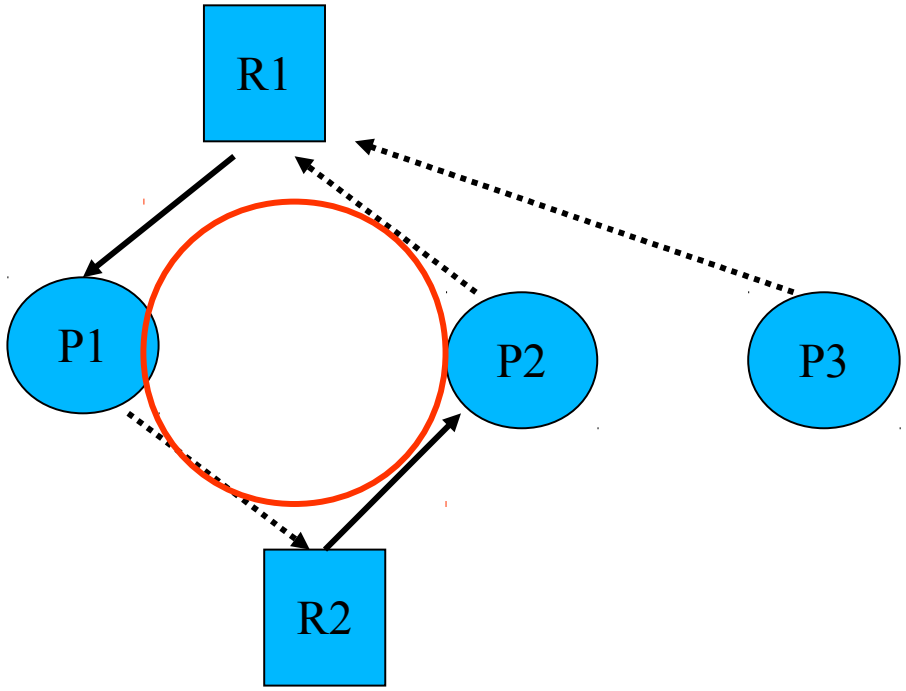
graphe des allocation/demande



Concurrence : Interblocage

graphe des allocation/demande

Détection de cycles -> interblocage

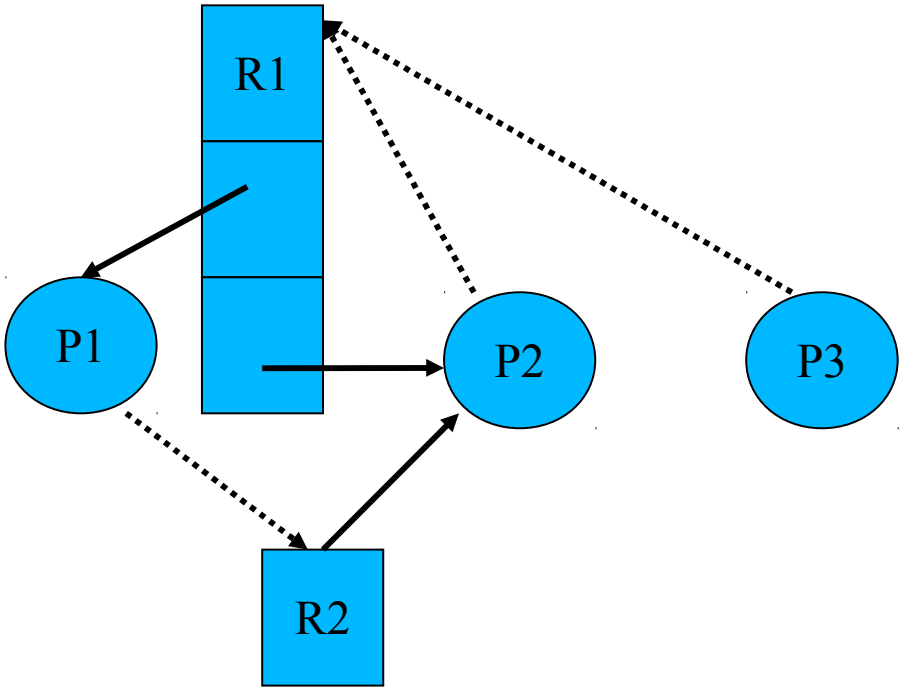


P1 et P2 sont bloqués et les autres processus comme P3 seront bloqués

Concurrence : Interblocage

graphe des allocation/demande

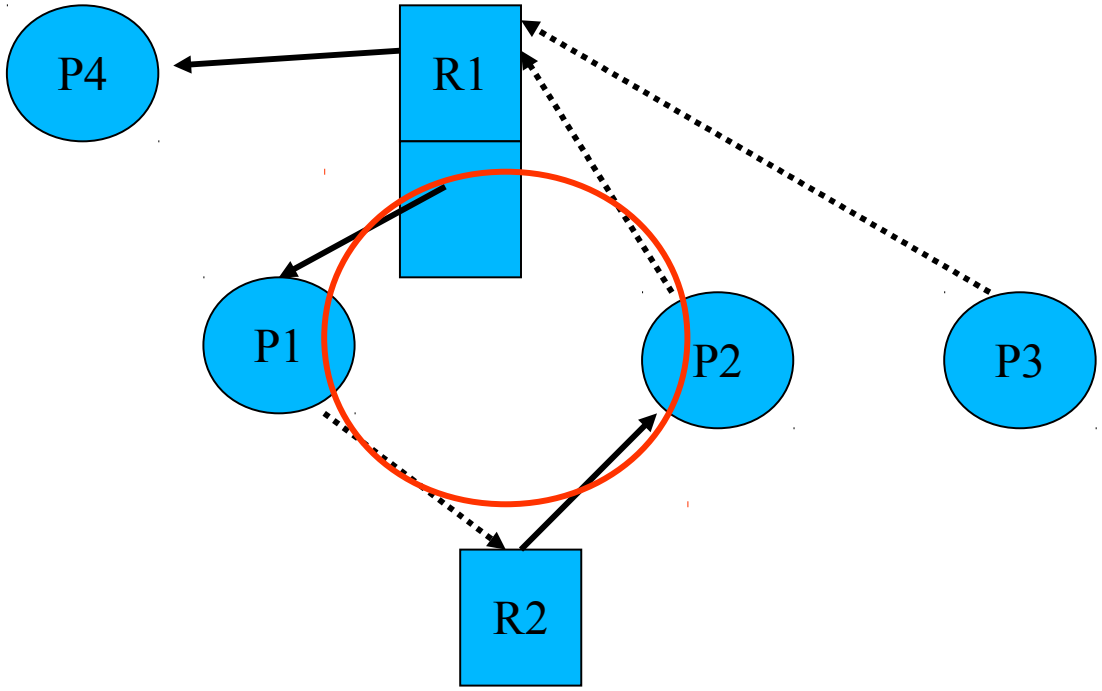
ressource à exemplaires multiples



Concurrence : Interblocage

graphe des allocation/demande

cycle mais pas d'interblocage

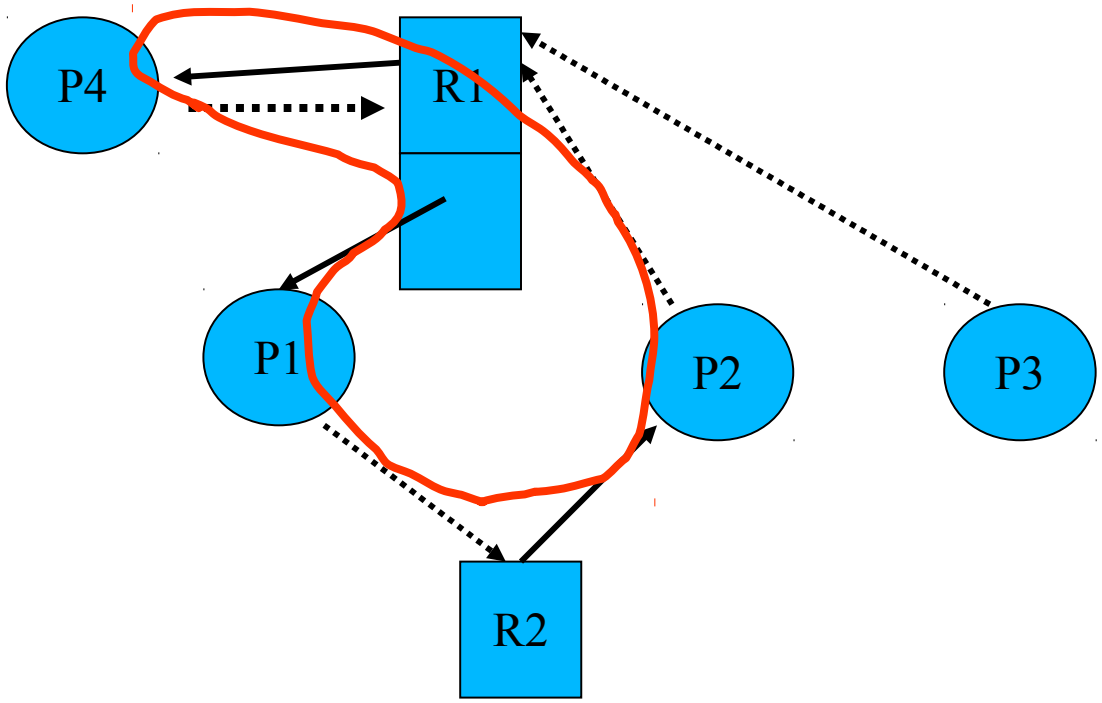


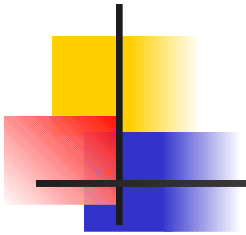
P4 devrait libérer R1 v1 qui sera allouée à P2 ou P3

Concurrence : Interblocage

graphe des allocation/demande

interblocage aucune flêche ne sort du cycle



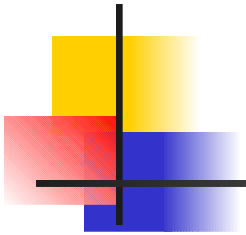


Concurrence : Interblocage

Traitement des interblocages

. prévention (refuse les ressources qui vont entraîner les interblocages)

. détection (On laisse les interblocages se produire, et le système recherche les interblocages)

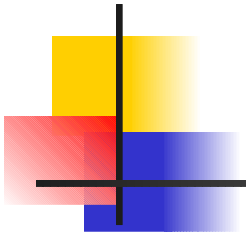


Concurrence : Interblocage

Traitement des interblocages

Curatif :

- **détection**
- **destruction un par un des processus (libération des ressources)**



Concurrence : Interblocage

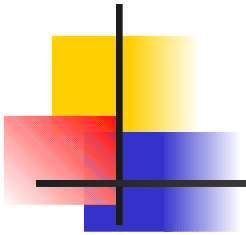
Traitement des interblocages

prévention (les politiques):

1) tout ou rien (le processus annonce les ressources nécessaires dès sa création et l'OS lui alloue que si elles sont toutes disponibles)

2) …

Concurrence : Interblocage



Traitement des interblocages

prévention (les politiques):

2) (dynamique) on numérote toutes les ressources R1 R2 RN

les processus demandent les ressources dont il a besoin dans l'ordre de numéro

R3 R10 R45 succes

R3 R10 R9 echec (refus)

exemple de deux processus

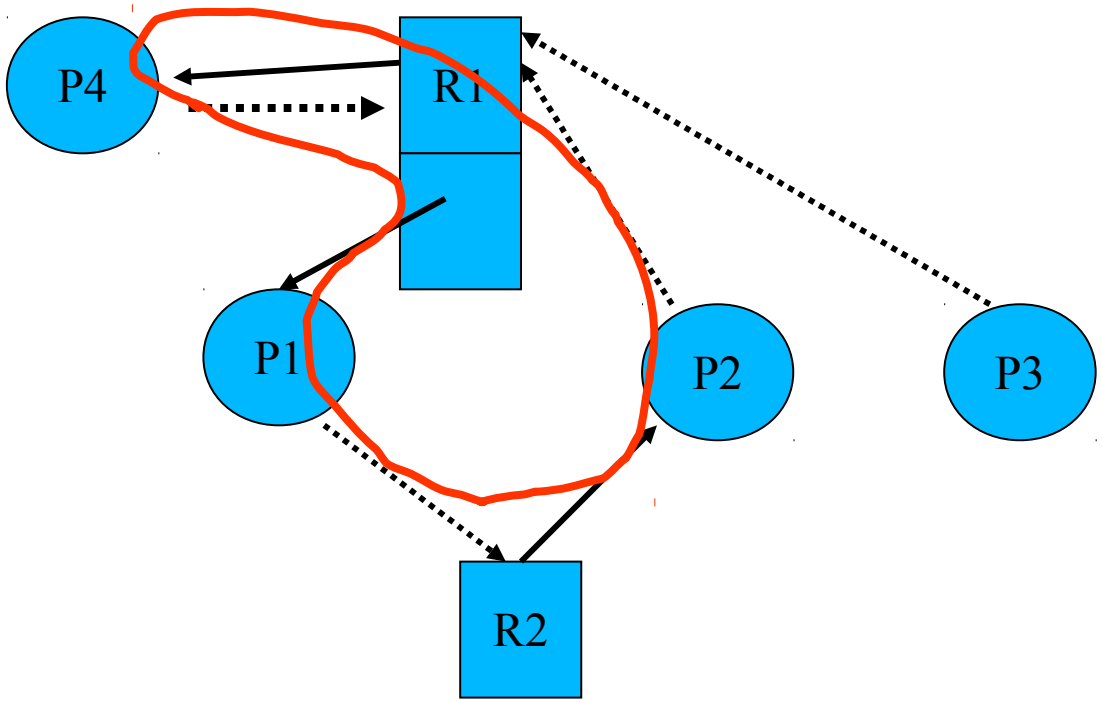
P1 R1 R2

P2 R2 R1 (echec)

P2 doit demander R1 puis R2 et donc attendre la fin de P1

Concurrence : Interblocage

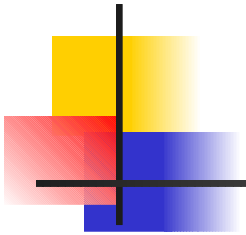
graphe des allocation/demande



P4 R1V1 R1V2 (ok)

P1 R1V2 R2 (ok)

P2 R2 R1v? (impossible)



Concurrence : Interblocage

Traitement des interblocages

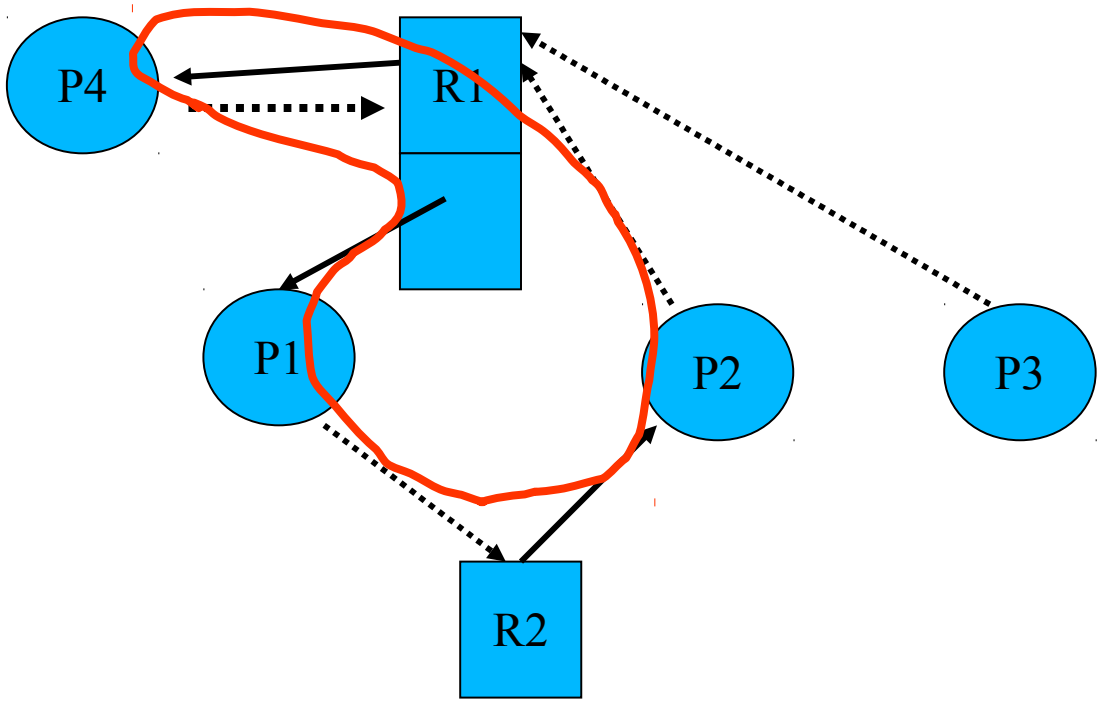
P1 possède R1 R10 R19 et désire R5

Dynamiquement pour obtenir une ressource de numéro inférieur à 19 il doit libérer les ressources de numéro supérieur à celle souhaitée

P1 libère de R10 et R19 et demande R5 R10 et R19

Concurrence : Interblocage

graphe des allocation/demande



P4 R1V1 R1V2 (ok)

P1 R1V2 R2 (ok)

P2 R2 R1v? (impossible) donc P2 libère R2 puis demande R1vX puis R2

Concurrence : Interblocage

Traitement des interblocages

Prévention : Algorithme du banquier

soit E le vecteur des ressources totales

	R1	R2	R3	R4
E	5	2	3	1

T la matrice des ressources détenues par chaque processus

T	R1	R2	R3	R4
P1	1	0	1	0
P2	2	0	0	1
P3	0	1	2	0

Concurrence : Interblocage

Traitement des interblocages

Prévention : Algorithme du banquier

R la matrice de demandes de chaque processus

R	R1	R2	R3	R4
P1	2	0	0	1
P2	1	0	1	0
P3	2	1	0	0

O le vecteur vide

	R1	R2	R3	R4
0	0	0	0	0

Concurrence : Interblocage

Algorithme du banquier

```
trouverAttenteCirculaire(E ressource,T détenue,R requete){
        D = E – somme(i=1,m) Tij  // ressources restantes
        while (   0 < Ri <= D)  {
            D += Ti  // restitue les ressources
            Ri = 0
        }
        return ( E != D)      donc un cycle   (si E ==D pas de cycle))
}
```

Concurrence : Interblocage

Algorithme du banquier

	R1	R2	R3	R4
E	5	2	3	1

T	R1	R2	R3	R4
P1	1	0	1	0
P2	2	0	0	1
P3	0	1	2	0

R	R1	R2	R3	R4
P1	2	0	0	1
P2	1	0	1	0
P3	2	1	0	0

D = E – somme(Ti) = **(5 2 3 1) - (3 1 3 1) D = (2 1 0 0)**

while (0 < Ri <= D)

R1 (2 0 0 1) > D (2 1 0 0)

R2 (1 0 1 0) > D (2 1 0 0)

R3 (2 1 0 0) < = D (2 1 0 0)

R 3 (0 0 0 0) et D += Tp3 (0 1 2 0) D= (2 2 2 0)

R1 (2 0 0 1) > D (2 2 2 0)

R2 (1 0 1 0) < = D (2 2 2 0)

R 2 (0 0 0 0) et D += Tp2 (2 0 0 1) D= (4 2 2 1)

Concurrence : Interblocage

Algorithme du banquier

	R1	R2	R3	R4
E	5	2	3	1

T	R1	R2	R3	R4
P1	1	0	1	0
P2	2	0	0	1
P3	0	1	2	0

R	R1	R2	R3	R4
P1	2	0	0	1
P2	1	0	1	0
P3	2	1	0	0

R1 (2 0 0 1) < = D (4 2 2 1)

R 1 (0 0 0 0) et D += Tp1 (1 0 1 0) D= (5 2 3 1)

return D= (5 2 3 1) != E (5 2 3 1) pas de cycle

Concurrence : Interblocage

Algorithme du banquier Autre exemple

	R1	R2	R3	R4
E	5	2	3	1

T	R1	R2	R3	R4
P1	1	0	1	0
P2	2	0	0	1
P3	0	1	2	0

R	R1	R2	R3	R4
P1	2	0	0	1
P2	3	0	1	0
P3	2	1	0	0

D = E – somme(Ti) = **(5 2 3 1) - (3 2 3 1) D = (2 1 0 0)**

while (0 < Ri <= D)

R1 (2 0 0 1) > D (2 1 0 0)

R2 (3 0 1 0) > D (2 1 0 0)

R3 (2 1 0 0) < = D (2 1 0 0)

R 3 (0 0 0 0) et D += Tp3 (0 1 2 0) D= (2 2 2 0)

R1 (2 0 0 1) > D (2 2 2 0)

R2 (3 0 1 0) > D (2 2 2 0)

return D= (2 2 2 0) != E (5 2 3 1) cycle

Concurrence : Interblocage

Algorithme du banquier

	R1	R2	R3	R4
E	5	2	3	1

T	R1	R2	R3	R4
P1	1	0	1	0
P2	2	0	0	1
P3	0	1	2	0

R	R1	R2	R3	R4
P1	2	0	0	1
P2	3	0	1	0
P3	2	1	0	0

	R1	R2	R3	R4
0	0	0	0	0

La prochaine étape aboutie a un dead lock !

il ne fallait pas arriver à cette matrice de T

Concurrence : Interblocage

Algorithme du banquier

Chaque processus déclare le Max de ressources dont il aura besoin

MAX	R1	R2	R3	R4
P1	3	2	3	1
P2	1	3	2	3
P3	2	2	2	2

La situation reste saine lorsque à une étape donnée même si tous les processus demandent les ressources dont ils ont signalées (Max) il n'y a pas d'inter-blocage

Le système peut allouer les demandes R pour cette étape.

Concurrence : Interblocage

Algorithme du banquier

```
situationSaine(E ressource,MAX , T détenue){
    return ! trouverAttenteCirculaire( E, T,MAX -T)
                                // déclaré maximun  - détenue  (reste)
}
```

Concurrence : Interblocage

Algorithme du banquier

```
AccepterRequete(E,Max,T,R) {
 while situationSaine(E,Max,T+Ri) {
     T += Ri // alloue les ressources
     Ri = 0
   }
}
```

Concurrence : Interblocage

Algorithme du banquier

	R1	R2	R3	R4
E	5	2	3	1

T	R1	R2	R3	R4
P1	2	1	3	1
P2	0	0	1	1
P3	0	2	1	0

MAX	R1	R2	R3	R4
P1	3	2	3	1
P2	1	3	2	3
P3	2	2	2	2

R	R1	R2	R3	R4
P1	0	0	0	0
P2	3	0	1	0
P3	2	1	0	0

traite R1 (1 0 1 1)
situationSaine(E,Max,T+ R1(1 0 1 1)) oui
situationSaine(E,Max,T+ R2(3 0 1 1)) oui
situationSaine(E,Max,T+ R3(2 1 0 0)) oui
satisfait P1

Concurrence : Interblocage

Algorithme du banquier

	R1	R2	R3	R4
E	5	2	3	1

T	R1	R2	R3	R4
P1	2	1	3	1
P2	0	0	1	1
P3	0	2	1	0

MAX	R1	R2	R3	R4
P1	3	2	3	1
P2	1	3	2	3
P3	2	2	2	2

R	R1	R2	R3	R4
P1	0	0	0	0
P2	3	0	1	0
P3	2	1	0	0

traite R2 (3 0 1 1)

situationSaine(E,Max,T+ R2(3 0 1 1)) non

situationSaine(E,Max,T+ R3(2 1 0 0)) non

Bloque P2 et P3 (sinon l'étape suivante peut engendrer un dead lock)

La libération des ressources de P1 va relancer l'algorithme