ED 7
Interblocage

Exercice 1 : Caractérisation de l'interblocage et prévention

On dispose d'un certain nombre de ressources critiques nécessaires à l'exécution de programmes : table traçante, imprimante, modem…
Chaque tâche, lors de son exécution, demande l'allocation de ressources lorsque celles-ci sont nécessaires. Les ressources sont libérées au bout d'un temps fini et au pire en fin d'exécution de la tâche.
Un exemple de programmation de trois tâches est le suivant ;

Tâche P1	Tâche P2	Tâche P3
demander(table_traçante)	demander(modem)	demander(imprimante)
demander(modem)	demander(imprimante)	demander(table_traçante)
--exécution	--exécution	--exécution
libérer(modem)	libérer(imprimante)	libérer(table_traçante)
libérer(table_traçante)	libérer(modem)	libérer(imprimante)

Question 1

Montrer qu'il y a risque d'interblocage pour ce comportement des tâches.

On analysera en particulier les quatre conditions nécessaires à l'apparition d'un interblocage.

Question 2

Proposer une solution de prévention statique de l'interblocage pour ce cas.

Exercice 2 : Analyse de condition d'interblocage

Un système comporte 11 ressources d'une classe de ressources banalisées (par exemple des blocs disque) ; 5 tâches se partagent ces ressources en les demandant une à une, chaque tâche ne peut demander plus de 3 ressources au total et restitue ses ressources au bout d'un temps fini.

Question 1

Montrer qu'il ne peut y avoir interblocage dans un tel système.

On généralise le problème précédent : N tâches se partagent M ressources d'une classe de ressources banalisées qu'ils demandent une à une, chaque tâche ne peut demander plus de T ressources. On a les relations suivantes :

T <M + 1 et N*T<M+N

Question 2

Montrer que l'interblocage est impossible dans un tel système.

On suppose maintenant que les 5 tâches se partagent 11 ressources d'une classe A de ressources banalisées, la demande maximale Ta pour chaque tâche est 3 et 6 ressources d'une classe B de resssources banalisées, la demande maximale Tb pour chaque tâche est 2. Les ressources sont acquises une à une et restituées au bout d'un temps fini.

Question 3

Montrer que contrairement au cas où il n'y a qu'une classe de ressources, il y a risque d'interblocage dans un tel système.

Question 4

Montrer que si le système dispose de 14 ressources de classe A et 7 ressources de classe B, il n'y a plus de risque d'interblocage.

Exercice 3 : Détection et prévention dynamique de l'interblocage

On considère un système comprenant 16 cases de mémoire (ressources banalisées) partagées par 3 processus(tâches).

A l'instant t1, on a l'état suivant :

processus	Allocation	Requête	Disponible
P0	5	0	0
P1	5	2	
P2	6	2	

A l'instant t2, on a l'état suivant :

processus	Allocation	Requête	Disponible
P0	5	1	0
P1	5	2	
P2	6	3	

A l'instant t3, on a l'état suivant :

processus	Allocation	Requête	Disponible
P0	3	1	3
P1	7	2	
P2	3	3	

Question 1

Pour les instants t1, t2 et t3, dire s'il y a ou non interblocage. On montrera que l'état du système est sain ou non.

On suppose que la demande maximale de chacun des processus est de 10 cases mémoire.

Question 2

Pour les instants t1, t2 et t3, dire s'il y a ou non risque d'interblocage . On montrera que l'état du système est fiable ou non.

Exercice 4 : Interblocage par verrouillage de parties d'un fichier

Sous Unix/Linux, il existe une fonction système appelée `fcntl`() qui permet de verrouiller et de déverrouiller une section (ou la totalité) d'un fichier. Cette fonction utilise la structure de données flock définie dans la librairie <`fcntl.h`>. La structure `flock` contient les champs suivants :

```
struct flock {
      short l_type ;    /* type: F_RDLCK, F_WRLCK ou F_UNLCK */
      short l_whence ;  /* portée: SEEK_SET, SEEK_CUR ou SEEK_END */
      off_t l_start ;   /* position par rapport à l_whence */
      off_t l_len ; /* longueur: si =0 <==> jusqu'à la fin du fichier */
      pid_t l_pid ;     /* pid du processus qui pose ce verrou */
}
```

Le champ l_type peut prendre l'une des valeurs suivantes :

- F_RDLCK : verrou de lecture, partagé par plusieurs processus,
- F_WRLCK : verrou en écriture, verrou exclusif (accès interdit par d'autres processus),
- F_UNLCK : déverrouillage.

Le verrouillage en écriture nécessite au préalable l'ouverture du fichier en écriture.

Le champ l_whence peut prendre une des valeurs suivantes pour définir la portée du fichier :

- SEEK_SET : à partir du début du fichier,
- SEEK_CUR : à partir de la position courante du curseur sur le fichier,
- SEEK_END : depuis la fin du fichier.

La portée commence à l_whence + l_start et sur l_len octets. Si l_len = 0, alors la portée va jusqu'à la fin du fichier et suit la fin du fichier si la taille de celui-ci augmente.

La fonction suivante :
fcntl(d, F_SETLKW, &verrou);
permet de poser un verrou sur une section d'un fichier de descripteur d. Le mode de verrouillage ainsi que la portée sont définis dans la variable verrou.
Si la section spécifiée dans la variable verrou est déjà verrouillée, il y a blocage du processus appelant jusqu'à déverrouillage de cette section.

Supposons que l'on dispose d'un fichier Fiche.txt qui dont le contenu est le suivant :
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ

Soient les opérations suivantes :
a- verrouiller en écriture une suite de 5 caractères du fichier Fiche.txt à partir du 10ème caractère.
b- verrouiller en écriture une suite de 5 caractères du fichier Fiche.txt à partir du 20ème caractère.

Supposons que le processus P1 après qu'il ait effectué l'opération a, fait un traitement quelconque qui prend 100ms avant de tenter de faire l'opération b, sachant que le processus P2 se contente de faire l'opération b suivi d'un traitement quelconque qui prend 200ms.
Le verrouillage sera possible même si c'est P2 qui commence à s'exécuter avant P1.

Question 1

Décrire le comportement de P1 et de P2 vis-à-vis des opérations de verrouillage si P1 démarre son exécution en premier.

Nous complétons le scénario précédent en supposant que P2 après l'opération b et le traitement de 200ms tente d'effectuer l'opération a.

Question 2

Quelle situation obtient-on dans ce cas ?

Sachant que le système refuse une opération de verrouillage qui présente un risque d'interblocage en retournant dans `fcntl()` *-1 avec* `EDEADLK` *dans* `errno`*, quel est le processus qui verra sa tentative de verrouillage échouée ?*