Partie 1

Exercices

1.1 Connaissances

1. Que se passe t-il au niveau matériel lors de l'execution d'un programme ?
2. Qu'est-ce qu'un processus ?
3. Représenter par un graphe les états élu, prêt, et bloqué. Ajouter l'état de démarrage et de fin (démarrer, terminer).
4. Définir les états élu, prêt, et bloqué.
5. Qu'est ce que l'ordonnanceur ?
6. D'après wikipedia, un **interblocage** peut survenir lorsque deux processus en concurrence pour deux ressources sont dans un ordre opposé. Expliquer comment l'interblocage peut survenir.
7. Représenter cette situation à l'aide d'un graphe :

- un arc de la ressource Ri au processus Pj signifie que le processus Pj a obtenu la ressource.
- un arc Pj vers Ri signifie que le processus Pj demande la ressource Ri.

On rappele qu'il y a interblocage lorsque des cycles sont présents dans ce graphe.

1.2 Exercice sur l'interblocage et graphes

Sept processus Pi sont dans la situation suivante par rapport aux ressources Ri :

- P1 a obtenu R1 et demande R2 ;
- P2 demande R3 et n'a obtenu aucune ressource tout comme P3 qui demande R2 ;
- P4 a obtenu R2 et R4 et demande R3 ;
- P5 a obtenu R3 et demande R5 ;
- P6 a obtenu R6 et demande R2 ;
- P7 a obtenu R5 et demande R2. On voudrait savoir s'il y a interblocage.

a. Construire un graphe orienté où les sommets sont les processus et les ressources, et où :

- la présence de l'arc Ri ⟶ Pj signifie que le processus Pj a obtenu la ressource Ri ;
- la présence de l'arc Pj ⟶ Ri signifie que le processus Pj demande la ressource Ri.

b. Il y a interblocage lorsque des cycles sont présents dans le graphe. Chercher ces cycles afin de déterminer s'il y a bien interblocage.

1.3 Exercice sur les tableaux en HTML (voir projet SQL web)

1.3.1 Tableaux HTML

Pour afficher un tableau dans une page HTML, on utilise à minima les balises :

- `<td>` pour les colonnes
- `<tr>` pour les lignes

Ces balises sont imbriquées de la manière suivante :

```
<table>
<tr><td>ID</td><td>nom</td><td>Pass</td></tr>
...
</table>
```

> Question 1 : Ecrire le code html qui affichera le tableau suivant :

id	nom	prenom	mot de passe
1	Deuf	John	ax2a
2	Dit	Alain	bx3a
3	Bombeur	Jean	cx4a

Après avoir fait une requête en SQL, ces données sont extraites de la base de données sous la forme d'une liste, stockée dans la variable `rows`

On en donne un extrait du contenu de cette variable :

```
[[1,'Deuf','John','as2a'],[2,'Dit',...],...]
```

> Compléter le script python qui va construire et servir le script html qui affichera le tableau rempli. Pour cela, le code html est mis dans une variable unique, de type str (les balises sont mises dans la chaine de caractères) :

```
table_affiche = '<table border=1><tr>'

table_affiche += '<td>ID</td><td>nom</td><td>Pass</td>'
table_affiche+= '</tr>'
for row in rows:
  ...
  ...
  ...
  ...
...
```

Partie 2

Correction exercices

On observe sur le graphe ainsi obtenu que P4 attend R3 qui est détenu par P5 donc P4 est bloqué le temps que P5 libère R3. Mais P5 qui attend R5 est lui-même bloqué, de même que P7 qui attend R2 et on revient à P4 qui détient R2 : nous sommes donc dans une situation d'interblocage.

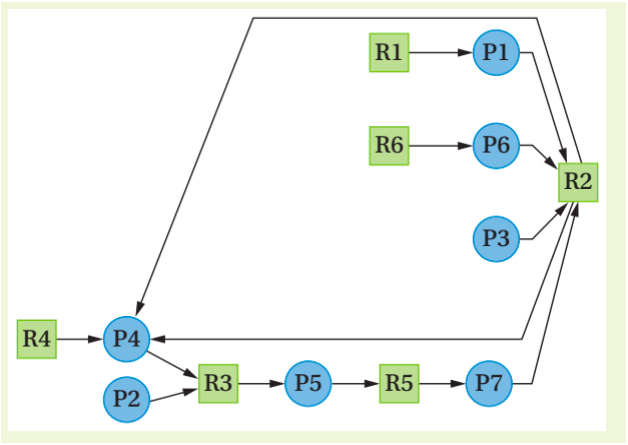

FIGURE 1 – graphe modelisant la situation

Partie 3

COURS

3.1 Définitions

Exécution d'un programme : le lancement d'un programme entraîne des lectures et écritures dans des registres et une partie de la mémoire. Le microprocesseur réalise des opérations sur les données mises dans les registres.

Un processus représente une instance d'exécution d'un programme dans une machine donnée.

3.2 Les états d'un processus

Les systèmes d'exploitation sont capables de gérer l'execution de plusieurs processus en même temps. En réalité, ce n'est pas tout à fait *en même temps* : pour gérer ce *chacun son tour*, les **systèmes d'exploitation** attribuent des *états* au processus.

Ces états sont résumés ci-dessous.

- Lorsqu'un processus est créé, il démarre dans l'état *prêt* : il attend de pouvoir avoir accès au processeur.
- Le processus obtient, grâce au systeme d'exploitation, l'accès au processeur. Il passe alors dans l'état *élu*.
- Alors qu'il est *élu*, le processus peut avoir besoin d'attendre une ressource quelconque comme, par exemple, une ressource en mémoire ou sur le disque dur. Il doit alors quitter momentanément le processeur pour que celui-ci puisse être utilisé à d'autres tâches (le processeur ne doit pas attendre !). Le processus passe donc de l'état élu à l'état *bloqué*. (c'est un *blocage*)
- Lorsque le processus a obtenu la ressource attendue mais s'est fait prendre sa place dans le processeur par un autre processus, il se met en attente : C'est l'état *prêt*, en attente que la *place se libère*. Cette étape, de bloqué à prêt est l'opération de *déblocage*.

- Le passage de l'état *prêt* vers l'état *élu* constitue l'opération *d'élection*.
- Un processus ne pourra *terminer* que s'il est déjà dans l'état *élu*.

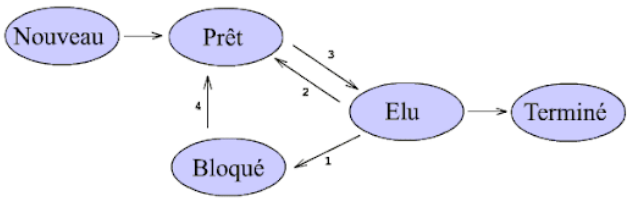

FIGURE 2 – mecanisme d'election

Un processus peut créer un ou plusieurs nouveaux processus. Il y a alors une filiation père-fils.

Chaque processus possède un numéro PID qui lui est attribué automatiquement. Il possède aussi un PPID qui est le numéro d'identification du processus père.

3.3 Ordonnancement

Plusieurs processus peuvent être dans l'état prêt : comment choisir celui qui sera élu ?

L'ordonnanceur (sheduler) classe les processus prêts dans une file et les fait passer du statut prêt à élu. Il planifie l'exécution des processus.

Dans les systèmes d'exploitation, l'ordonnanceur désigne le composant du noyau du *système d'exploitation* choisissant l'ordre d'exécution des processus sur les processeurs d'un ordinateur. Cette organisation permet d'occuper au mieux le (ou les) processeurs.

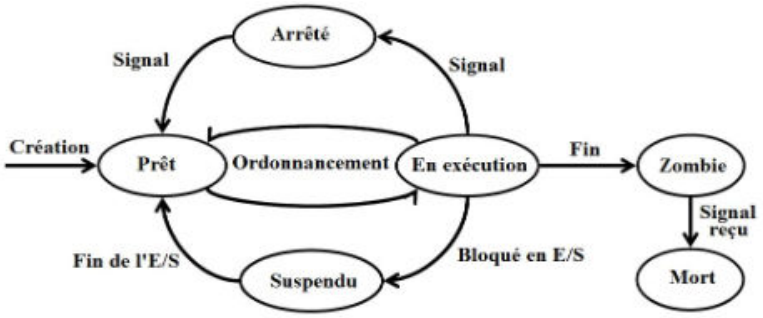

FIGURE 3 – gestion des processus

Il existe plusieurs politiques d'ordonnancement dont le choix va dépendre des objectifs du système. Voici quelques exemples :

• Premier arrivé, premier servi : simple, mais peu adapté à la plupart des situations.
• Plus court d'abord : très efficace, mais il est la plupart du temps impossible de connaître à l'avance le temps d'exécution d'un processus.
• Priorité : le système alloue un niveau de priorité aux processus (SCHED_FIFO sur Linux). Cependant des processus de faible priorité peuvent ne jamais être élus.
• Tourniquet : un quantum de temps est alloué à chaque processus (SCHED_RR sous Linux). Si le processus n'est pas terminé au bout de ce temps, il est mis en bout de file en état prêt.
• Un système hybride entre tourniquet et priorité qu'on retrouve dans les systèmes Unix.

Partie 4

Interblocage

4.1 Concurents mal synchronisés

L'ordonnanceur fait passer les processus de *élu* à *commuté*. Mais Il peut y avoir une situation où 2 processus sont **interbloqués** car :

- P1 possède la ressource R1 mais souhaite R2 : commutation pour P2
- P2 possède la ressource R2 mais souhaite R1 : commutation pour P1

Dans cette situation, les deux processus légers sont définitivement bloqués.(wiki)

Des solutions de détection/guérison peuvent être mises en place.

4.2 Définition sur Wikipedia

Deux processus en concurrence pour deux ressources dans un ordre opposé. Voici une chronologie possible qui mène à un interblocage.

A) Un seul processus se déroule.

B) Le processus ultérieur doit attendre.

C) Un blocage se produit lorsque le premier processus verrouille la première ressource en même temps que le second processus verrouille la seconde ressource.

D) Le blocage peut être résolu en annulant et en redémarrant le premier processus.

Exemple : le processus P1 utilise la ressource R2 qui est attendue par le processus P2 qui utilise la ressource R1, attendue par P1.

4.3 Modélisation par un graphe orienté

Il y a interblocage lorsque des cycles sont présents dans le graphe réalisé de la manière suivante :

- un arc de la ressource Ri au processus Pj signifie que le processus Pj a obtenu la ressource
- un arc Pj vers Ri signifie que le processus Pj demande la ressource Ri.

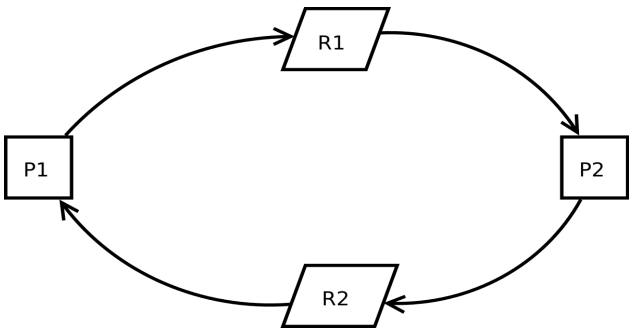

Figure 4 – interblocage - wikipedia

Partie 5

TP en console

5.1 La commande ps

Pour la liste des processus obtenue par `ps`

- PID : numéro du processus (processus identifier)
- PPID : identifiant du parent qui a engendré le processus
- TTY
- TIME
- CMD : commande qui a lancé le processus

5.2 La commande top :

Cette commande est l'équivalent du gestionnaire de tâches de Windows. Elle apporte donc des renseignements sur la consommation mémoire, CPU, buffer et tous les processus en cours. Son intérêt est qu'elle apporte des statistiques de consommation en temps réel.

- PID : numéro du processus
- USER : utilisateur qui fait tourner le process
- %CPU : la consomation du CPU
- %MEM : la consomation de la RAM
- TIME+ : le temps d'utilisation CPU depuis que le process est lancé
- COMMAND : le processus en lui-même

unix	windows	commande
`ps`	`tasklist / svc`, `tasklist /?` (processus)	liste des processus
`ps -aef`	`tasklist / svc`, `sc /?` (services)	liste des services
`kill <PID>` ou `kill <PPID>`	`taskkill <nom>`(sortie avec `q`	fermer un processus, directement ou avec le PID du parent
`top`		suivi en temps réel des processus

Sous windows, on peut aussi utiliser le Task manager

Partie 6

Notes et ressources

-
- si vous souhaitez un environnement Windows semblable à Linux, vous pouvez utiliser **Cygwin**. Il apporte l'environnement Linux à Windows. Vous pouvez utiliser presque toutes les commandes.