Université A. Mira Bejaia 14/ 03/2015
Faculté des Sciences Exactes Durée 2 h 00
Département d'Informatique
Année d'étude : 3eme Année Licence Académique

Examen de Systèmes d'Exploitation 2

Remarques importantes :
- L'**exercice1** et une partie des **Questions de cours** compteront comme 2eme interrogation.
- Les parties 1 et 2 des exercices 2 et 3 sont indépendantes.

Exercice 1 : (*Parallelisation d'un système de tâches*) (5pts)

Remarque : les questions 1), 2) et 3) sont indépendantes.

1) On considère le programme utilisant les primitives parbegin/parend donné ci-dessous:

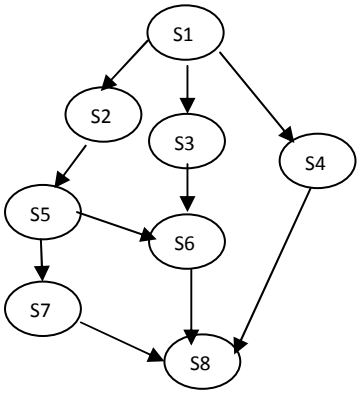
```
begin
        parbegin
                lire(a)
                lire(b)
        parend ;
        parbegin
                c := a*a
                begin
                        d := a*b ;
                        e := d*a ;
                end
        parend ;
        e := c + d + e
end
```

- a) Donner le graphe de précédence en précisant les tâches que vous considérez.

-b) Indiquer les domaines de lecture et écriture des différentes tâches.

2) Donner un exemple simple de système de tâche (*sous forme d'un graphe de précédence*) ne pouvant pas être décrit avec les primitives parbegin/parend.

3) soit le graphe de précédence suivant :



Donner le programme parallèle correspondant en utilisant les primitives fork/join.

<u>Exercices 2</u> : Gestion de la mémoire (6.5pts)

Remarque : Les parties 1 et 2 sont indépendantes

<u>Partie 1 :</u> Etant donnés :

- Une table des pages de taille 128 Ko
- Le nombre d'entrées de la table des pages égal à 65536
- Le déplacement dans la page est codé sur 16 bits
- Une entrée de la table des pages est de la forme : |n|1|3| où
    - n est nombre de bits pour coder un cadre de page (*une case*)
    - 1 est le bit d'absence/présence
    - 3 est le nombre de bits pour coder la date de chargement de la page

➔ Chaque entrée de la table contient donc n+4 bits. On vous demande de (*vous devez détailler votre réponse*) :

1) Déterminer la valeur de n.
2) Calculer la taille de la mémoire physique.
3) Donner la taille de l'espace d'adressage logique
4) Dire si le nombre d'entrées de la table des pages change si on augmente la taille de la mémoire physique de 1 Mo. Si oui, de combien augmente-il ?

<u>Partie 2 :</u>

Soit un processus qui dispose de huit pages logiques (*numérotées de 0 à 7*) et qui lui alloué trois pages physiques (*initialement vides*). La séquence de demande de pages est la suivante :

0-1-2-3-0-1-2-0-3-0-2-3-4-5-6-7

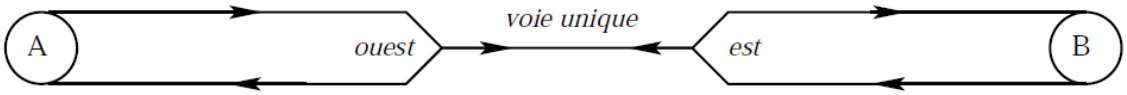
Donner le nombre de défauts de pages produit par chacun des algorithmes suivants (*en précisant les différentes étapes*):

a) LRU (*Least Recently Used ou moins récemment utilisé*),
b) MRU (*Most Recently Used ou Plus récemment utilisée*),
c) LFU (*Least Frequently used ou Moins fréquemment utilisée*). Dans le cas d'égalité de fréquences alors l'algorithme FIFO est appliqué.

Exercices 3 : Synchronisation (5.5pts)

Partie 1 : (2.5pt) (Synchronisation par moniteurs)

Une ligne de chemin de fer reliant deux villes A et B comporte une section à voie unique (figure). On représente les trains par des processus, dont l'algorithme général est décrit ci-après :



Processus A_vers_B

```
Voie_unique.entrée_ouest()
    <trajet voie unique>
Voie_unique.sortie_est()
```

Processus B_vers_A

```
voie_unique.entrée_est()
    <trajet voie unique>
voie_unique.sortie_ouest()
```

Le rôle du moniteur voie unique est de garantir que tous les trains engagées à un instant donné sur la voie unique circulent dans le même sens.

Question) Que pouvez-vous dire sur cette variante du moniteur gérant une voie_unique de capacité maximum N ? Discuter selon les 2 dimensions : **risque de famine ou non**, et **occupation optimale ou non des N places sur la voie**. Argumentez votre réponse en étudiant les différents scénarios possibles.

Moniteur voie_unique /*avec limite*/
// Variables communes :
Condition attend_ouest, attend_est;
Entier nb_engages_ouest = 0 ;
nb_engages_est = 0 ;

```
Entre_ouest / (symétriquement entrée_est)
{
   if (nb_engages_est >0) OR
     (nb_engages_ouest = N) OR
      (attend_est<> vide) then
                  attend_ouest.Wait;
   End if
    nb_engages_ouest++;
    if attend_est =vide  then
        if nb_engages_ouest<N
            then attend_ouest.signal ;
        endif
    endif
}
```

```
Sortie_est/  (symétriquement sortie_ouest)
{
   nb_engages_ouest --;
   if (nb_engages_ouest = 0) then
       if (attend_est <> vide) then
                (attend_est).signal;
           Else (attend_ouest).signal;
       Endif
   Else
      attend_ouest.signal ;
   endif
}
```

Partie 2 : (par sémaphores) (3pts)

Soit l'exécution parallèle des deux processus suivants :

Debut
Parbegin ProcessusA ; ProcessusB ; **Parend**
Fin

```
ProcessusA :
Debut
        Faire toujours
                T1 ;
        Fait
fin
```

```
ProcessusB :
Debut
        Faire toujours
                T2 ;
        Fait
fin
```

1) Utilisez un sémaphore pour synchroniser les 2 processus de telle manière que l'exécution de la tâche T1 ne soit jamais simultanée avec l'exécution de la tâche T2.

2) Utilisez deux sémaphores pour synchroniser les 2 processus de telle manière que les tâches se déroulent toujours dans l'ordre : T1T2T1T2T1T2...

3) Utilisez deux sémaphores pour synchroniser les 2 processus de telle manière que les tâches se déroulent toujours dans l'ordre : T1T2T2T1T2T2T1T2T2...

Question de Cours (3 pts)

Question 1) Quels sont les avantages d'un sémaphore par rapport à un outil du type «test and set» ? Justifier.

Question 2) : On propose les fonctions d'exclusion mutuelle suivantes, écrites en C, pour un système à deux processus:

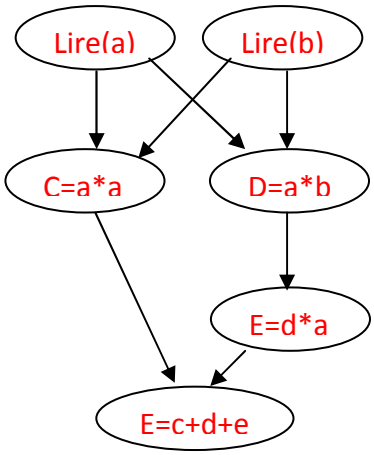
```
int[] sc = {0,0};
void entrer_section_critique(int id) {
sc[id]=1;
while(sc[1-id]){};
}
void sortir_section_critique(int id) {
sc[id]=0;
}
```

1) Expliquez quel est le problème de cette solution. Illustrez sur un exemple concret.

2) Proposez une solution simple pour éviter de problème.

Bonne Chance

Corrigé de l'examen



Exercice 1 : (*Parallelisation d'un système de tâches*) (5 pts)

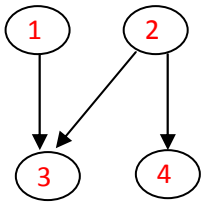
1.a) graphe de précédence (1pt)

1.b) les domaines de lecture et écriture des différentes tâches. (**1.5** Pts)
(0.25pts Pour chaque deux domaine correct).

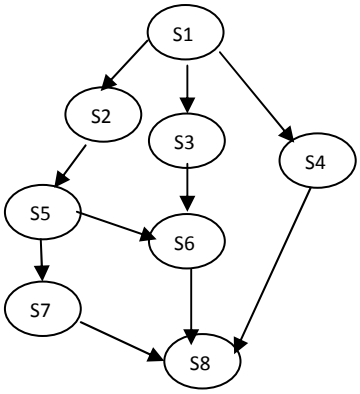
taches	Domaine de lecture(L)	Domaine d'ecriture (E)
Lire(a)	{ }	{a}
Lire(b)	{ }	{b}
C=a*a	{a}	{c}
D=a*b	{a,b}	{d}
E=d*a	{a,d}	{e}
E=c+d+e	{c,d,e}	{e}

3) Exemple simple de système de tâche ne pouvant pas être décrit avec les primitives parbegin/parend.

Réponse (**0.5 pt**): (Voir le support de Cours ou le TD)



4) Soit le graphe de précédence suivant :




```
N1=2, n2=3;
S1 ;
Fork L1;
S2 ;
S5;
Fork L3;
S7;
Goto L4;
```

```
L1: fork L 2;
S3;
L3: join n1;
S6;
Goto L4;
L2: S4;
L4: join n2;
S8;
```

(**2 pts**) Donner le programme parallèle correspondant en utilisant les primitives fork/join.

Exercices 2 : Gestion de la mémoire (6.5pts)

Partie 1 : (3.5pt)

1) On a le nombre d'entrées de la table des pages égal à 65536 (ou 2^{16}), c'est-à-dire chaque entrée de la table est codée sur 16 bits. Comme une entrée de la table des pages est composée de (n+1+3) bits alors (3+1+3)=16 d'où **n=12** bits. (**1pt**)

2) La taille de l'espace physique est égal au nombre de pages physique multiplier par la taille d'une page.

   On a n=12 bit pour coder le nombre de page physique (ou cadre de page), alors le nombre de cadre de pages qu'on peut avoir est = 2^{12}, et la taille d'une page est égale à 2^{16} bits (donnée), alors la taille totale de l'espace physique est égal à : $2^{12} * 2^{16} = \mathbf{2^{28}\ bits.}$
   C'est-à-dire ($2^{28}/2^3 = 2^{25}$ Octets, ou $2^{25}/2^{10} = 2^{15}$ K0, ou $2^{15}/2^{10} = 2^5 =$ **32 Mo**). (**0.75pt**)

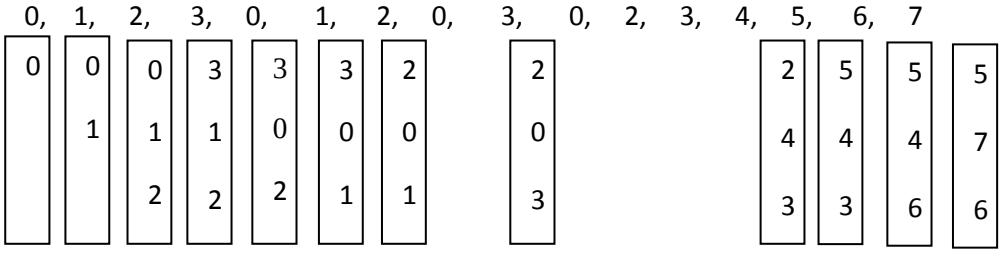
3) Etant donnée que le nombre d'entées de la table des pages est de 2^{16} alors logiquement, on peut coder jusqu'à 2^{16} pages logiques. Comme la taille d'une page est égale à 2^{16} alors la taille totale de l'espace logique est égale à : $2^{16} * 2^{16} = \mathbf{2^{32}\ bits.}$
   C'est-à-dire ($2^{32}/2^3 = 2^{29}$ Octets, ou $2^{29}/2^{10} = 2^{19}$Ko, ou $2^{19}/2^{10} = 2^9 =$ **512 Mo**). (**0.75pt**)

4) (**1 pt**) Quand on augmente la taille de la mémoire physique de 1 Mo, le nouveau espace physique passe de : 32 Mo vers 33 Mo. Cet espace n'est pas suffisant pour contenir l'espace logique qui est de 512 Mo.

   Le nombre d'entrées de la table des pages change (augmente) si *n* augmente. Pour des pages 2^{16}, si *n* augmente de 1 c'est-à-dire 12+1=13, alors il faut au moins un espace physique de 64 Mo ($2^{13} * 2^{16}$ bits $= 2^{9*}\ 2^{20} = 2^6 * 2^{20} = 64$Mo). Or le nouvel espace physique est de 33 Mo qui est inferieur à 64 Mo. D'où le nombre d'entrées de la table **ne change pas**.

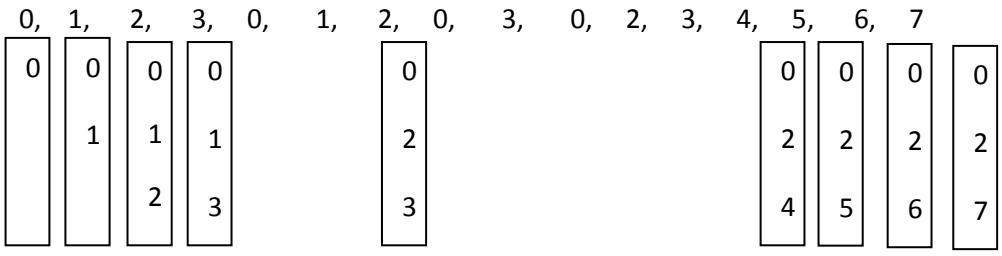
Partie 2 : (3 pt)

a) LRU (*Least Recently Used ou moins récemment utilisé*),



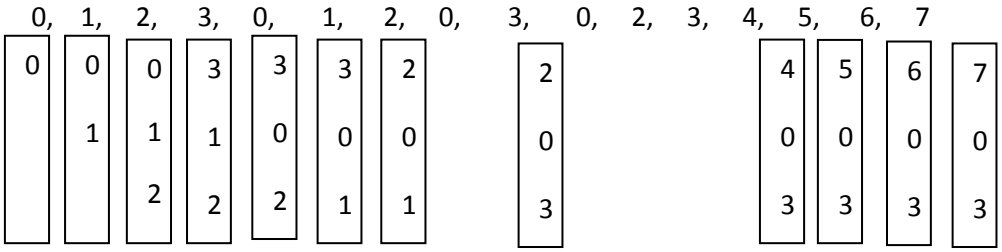
Il y'a 12 défauts de pages (**1 pts**)

b) MRU (*Most Recently Used ou Plus récemment utilisée*),



Il y' a 9 défauts de pages (1 pts)

c) LFU (*Least Frequently used ou Moins fréquemment utilisée*). Dans le cas d'égalité de fréquences alors l'algorithme FIFO est appliqué.



Il y'a 12 défauts de pages (1 pts)

Exercice 3 :

Partie1 : (synchronisation à l'aide des moniteurs) (2.5 pts)

Vis-à-vis du risqué de famine, On remarque que quand un train qui quitte la voie_unique si ce n'est pas le dernier d'un lot de trains dans le même sens, alors il peut passer son «slot libéré » à un train dans le même sens que lui. Ainsi, C**et algorithme ne résout pas bien la famine** (on ne sait pas borner le nombre de trains dont il faudra attendre le passage) = il essaye plutôt d'occuper la place libre de plus. (1.5 Pts)

Concernant l'utilisation des N places sur la voie, cette solution peut dégénérer à un cas « **un train dans chaque sens**, à tour de rôle », des lors que chaque fois qu'un train se présente dans un sens, il y'en a un autre en attente dans l'autre sens ! (1 pt)

Partie 2 : (Synchronisation à l'aide des sémaphores) (3pts)

1) (1pt) Utilisez un sémaphore pour synchroniser les 2 processus de telle manière que l'exécution de la tâche T1 ne soit jamais simultanée avec l'exécution de la tâche T2.

Sémaphore s=1 ;

```
ProcessusA :
Debut
        Faire toujours
        P(s) ;
        T1 ;
        V(s) ;
        Fait
fin
```

```
ProcessusB :
Debut
        Faire toujours
        P(s) ;
        T2 ;
        V(s) ;
        Fait
fin
```

2) (1 pts) Utilisez deux sémaphores pour synchroniser les 2 processus de telle manière que les tâches se déroulent toujours dans l'ordre : T1T2T1T2T1T2...

Sémaphore s1=1 ; */*pour permettre au processus A de commencer*/*
s2=0 ; */*pour empêcher le processus B de commencer jusqu'à ce que A le réveil */*

```
ProcessusA :
Debut
        Faire toujours
        P(s1) ;
        T1 ;
        V(s2) ;
        Fait
fin
```

```
ProcessusB :
Debut
        Faire toujours
        P(s2) ;
        T2 ;
        V(s1) ;
        Fait
fin
```

3) (1 pt) Utilisez deux sémaphores pour synchroniser les 2 processus de telle manière que les tâches se déroulent toujours dans l'ordre : T1T2T2T1T2T2T1T2T2...

```
Sémaphore  s1=2 ;   /*pour permettre au processus A de commencer*/
                    s2=0 ;   /*pour empêcher le processus B de commencer jusqu'à
                             ce que A le réveil */
```

```
ProcessusA :
Debut
        Faire toujours
        P(s1) ;  /* on attend deux signaux de B pour passer*/
        P(s1) ;
        T1 ;
        V(s2) ;  /* on donne à B deux signaux pour lui
        V(s2) ;   /* permette de passer deux fois*/
        Fait
fin
```

```
ProcessusB :
Debut
        Faire toujours
        P(s2) ;
        T2 ;
        V(s1) ;
        Fait
fin
```

Question de Cours (3 points=1+1+1)

Q1) Quels sont les avantages d'un sémaphore par rapport à un outil du type «test and set» ? Justifier.

Réponse : *Avec l'outil Sémaphore on n'a pas* ***d'attente active*** *(ou le processus passe à l'état bloqué dans une* ***fille d'attente****.*

Q2.1) Expliquez quel est le problème de cette solution. Illustrez sur un exemple concret.

Réponse : *Rien n'empêche deux processus de se marquer tous les deux en SC et donc de rentrer dans une boucle d'attente infinie. P0 appel le SC et effectue sc[0]=1, puis est interrompu, P1 fait de même, donc attend P0, qui reprend la main et attend P1...*

Q 2.2) Proposez une solution simple pour éviter de problème.

Réponse : *on peut rajouter une variable de tour de priorité. Lors de l'entrée en SC, on donne le tour à l'autre puis on met son booléen SC à vrai. On attend alors tant que à la fois l'autre à la main et il est en SC. Si l'autre fait pareil, il nous donnera la main et donc on n'attendra plus.*