Epreuve de Systèmes d'exploitation
Durée : 1heure
Tout document interdit

Instructions au candidat (à lire avant le début de l'épreuve)

- Les candidats doivent vérifier que le sujet comprend 3 pages.
- Les candidats doivent rendre les copies même vierges.
- Si au cours de l'épreuve, un candidat repère ce qui lui semble être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

Exercice1 : Création de Processus Unix (6 points)

Considérons le programme suivant :

```
int main()
{
   ( fork() || fork() ) && ( fork() || fork() );
/* (Fork1 || fork2 ) && (fork3 || fork4 ); Commentaire pour numéroter les fork() */
   printf("\n Père : %d; Fi ls : %d", getppid(), getpid());
   exit(0) ;
}
```

On suppose que le PID Shell est 2000, le PID du processus correspondant à ce programme est 2400 et l'algorithme d'ordonnancement est FIFO(ou FCFS), donner les résultats correspondant à l'exécution du programme ci-dessus.

Exercice2: Partitionnement MBR basé sur le BIOS : (4 points)

a) Donner le nombre maximum de partitions que l'on peut *utiliser* en même temps sur un système Windows.
b) Quelle est la taille maximale d'une partition(puissance de 2 : 2^n unités) ;
c) Donner le nombre de tables de partitions d'un disque dur (la ou les tables + éventuellement la ou les tables de sauvegarde) ;
d) Sur une machine on désire installer trois systèmes linux(Ex: Redhat, Ubuntu, Debian) et des systèmes Windows: Donner le nombre maximum de systèmes Windows que l'on peut installer sur cette machine.

Exercice3: Exclusion Mutuelle par attente active avec instruction spéciale (2points)

On cherche à réaliser l'exclusion mutuelle pour l'accès à une ressource critique par attente active en utilisant la fonction booléenne TMB dont la syntaxe est: TMB(Reg,M), et dont l'algorithme est:

```
Début
    Bloquer l'accès à la cellule mémoire M;
    Si M>Reg  Alors Début
                        M:= - (Reg);
                        Reg :=M ;
                Fin;
    Libérer l'accès à la cellule mémoire M;
Fin.
```

Avec Reg étant un registre, et M un mot mémoire.

Les numéros des processus sont toujours positifs et vont de 1 à PMAX. On désire savoir en lisant M si la ressource critique est libre, et si elle est occupée, quel est le numéro du processus qui l'occupe.

Questions:

On désire programmer l'Exclusion Mutuelle par attente active à l'aide de TMB selon le protocole classique d'entrée et de sortie de la section critique sachant que la fonction GetPid() permet à tout processus de récupérer son numéro.

Q1 : Choisir la bonne solution parmi les Solutions proposées:

(Réponse juste : +1,5 point ; Réponse fausse : - 0,5point ; Absence de réponse : 0 point)

SolutionA. M init (0)

1) Demande d'entrée : Reg := - (GetPid()) ;
TantQue (Reg<0) Faire TMB(Reg,M) ; FinFaire ;
2) Section Critique : <SC> ;
3) Sortie : M := 0 ;

SolutionB. M init -(1)

1) Demande d'entrée : Reg := - (GetPid()) ;
TantQue (M<=Reg) Faire TMB(Reg,M) ; FinFaire ;
2) Section Critique : <SC> ;
3) Sortie : M := - (1);

SolutionC. M init (Pmax+1) ;

1) Demande d'entrée : Reg := GetPid() ;
TantQue (Reg>0) Faire TMB(Reg,M); FinFaire ;
2) Section Critique : <SC> ;
3) Sortie : M := Pmax+1;

SolutionD. M init (Pmax)

1) Demande d'entrée : Reg := GetPid() ;
TantQue (Reg<M) Faire TMB(Reg,M); FinFaire ;
2) Section Critique : <SC> ;
3) Sortie : M := Pmax ;

SolutionE. M init (Pmax)

1) Demande d'entrée : Reg := - (GetPid()) ;
TantQue (Reg<=M) Faire TMB(Reg,M); FinFaire ;
2) Section Critique : <SC> ;
3) Sortie : M := Pmax ;

Q2 : Quelle est la valeur de M si la ressource critique est occupée par le processus de numéro K.

(Réponse juste : +0,5 point ; Réponse fausse ou Absence de réponse : 0 point)

Exercice4: Pont à voie unique (8 points).

Soit un pont à voie unique qui ne peut être traversé que dans un sens à la fois (sens i ;i=1 ou 2) . Tout véhicule (processus) de poids x kilos et désirant traverser le pont dans le sens i devra respecter le protocole suivant:

DTi (x): demande de traversée en sens i;
<traversée en sens i>;
FTi(x) : fin de traversée en sens i;

Le pont ne peut supporter une charge supérieure à Cmax kilos (on supposera tout x <= Cmax en entrée).
On désire imposer à tous les véhicules un passage en FIFO strict.

***Question:**

On veut programmer les procédures DTi(x) et FTi(x) en n'utilisant que les variables et sémaphores suivants :

Var Charge:integer init 0 ; /*indique la charge courante du pont (somme des poids des véhicules engagés sur le pont)*/ ;
FIFO :semaphore init 1 ; /* impose le passage des véhicules en fifo strict sur le pont*/
Sens : array [1..2] of semaphore init 1 ; /* Sens[i] permet d'autoriser ou bloquer le passage en sens i */
Mutexcharge : semaphore init 1 ;/* EM pour l'accès à la variable critique Charge */
Scapacité : semaphore init 0 ;/* attente du véhicule courant si risque de dépassement de capacité */
AC : integer init 0;/*mémorise le poids x du véhicule en attente de passage sur le pont s'il y a risque de dépassement de capacité */

```
Procedure DTi( x : integer) ;
Begin
      P(FIFO) ;
      P(Sens[i]) ;
      P(Mutexcharge) ;
      If(<DT1>) then <DT2> ;
      If((<DT3>)then begin
                  AC :=x ;
                  V(Mutexcharge) ;
                  P(Scapacite) ;
                  end ;
      Else begin
                <DT4> ;
                V(Mutexcharge) ;
         end ;
      V(Sens[i]) ;
      V(FIFO) ;
End DTi ;
```

```
Procedure FTi( x :integer) ;
Begin
      P(Mutexcharge) ;
      Charge :=Charge-x ;
      If(AC > 0) then begin
                  If((<FT1>) then begin
                        <FT2> ;
                        AC :=0 ;
                        V(Scapacite) ;
                        end ;
      end ;
      Else if (<FT3>) then <FT4> ;
      V(Mutexcharge) ;
End FTi ;
```

Compléter la Solution Proposée en remplaçant les conditions et les instructions manquantes
(8 réponses à fournir -→Réponse juste : +1 point ; Réponse fausse ou Absence de réponse : 0 point)
On donne:

a)Conditions à choisir :

C1=(Charge>Cmax)	C2=(Charge + AC > Cmax)	C3=(Charge + AC <= Cmax)	C4=(Charge>x)
C5= (Charge=0)	C6=(Charge<Cmax)	C7=(Charge+x<=Cmax)	C8=(Charge+x>Cmax)
C9= ((Charge+x>Cmax) or (AC>0))		C10= ((Charge+x>Cmax) and (AC>0))	

b)Instructions à choisir :

I1 : Charge :=Charge+1	I2 : Charge :=Charge-1	I3 : Charge :=Charge+x	I4 : Charge :=Charge-x
I5 : AC :=AC+x	I6 : AC :=AC-x	I7: Charge :=Charge+AC	I8: Charge :=Charge-AC
I9 : P(Mutexcharge)	I10 :V(Mutexcharge)	I11: P(Sens[i])	I12:P(Sens[3-i])
I13 : V(Sens[i])	I14 : V(Sens[3-i])	I15 : P(Scapacite)	I16: V(Scapacite)
I17 : P(FIFO)	I18 : V(FIFO)		