Partiel LI324

L3 – Licence d'Informatique Mars 2013

Tout document papier autorisé

Barème donné à titre indicatif

I. ORDONNANCEMENT ET PROCESSUS (11 POINTS)

Dans cet exercice on s'intéresse à l'exécution du programme suivant, en supposant que l'appel au fonction f() et g() génère respectivement des calculs de 60ms et 40ms et que l'entrée/sortie provoquée par l'écriture sur le terminal (printf) dure 30 ms.

Pour simplifier vous considérerez qu'il y a deux contrôleurs d'E/S et qu'il n'y a pas de commutation au moment du fork().

```
int main(int argc, char**argv) {
    f();
    int p=fork();
    f();
    printf("%d ", p);
    g();
    return (EXIT_SUCCESS);
}
```

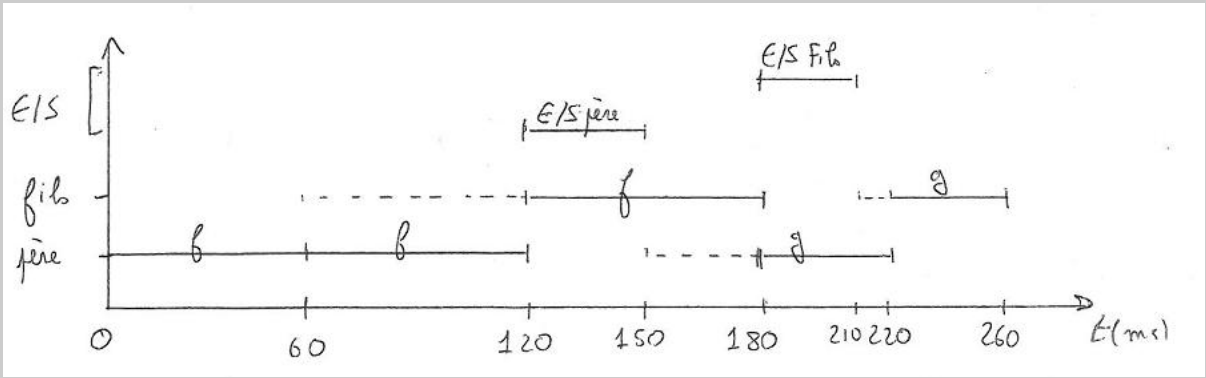
I.1. (2,5 points)

En considérant successivement un système fonctionnant en mode "batch" donnez :

a) le diagramme de Gantt

b) l'affichage sur le terminal

b) le taux d'utilisation du processeur sous forme d'une fraction non simplifiée.

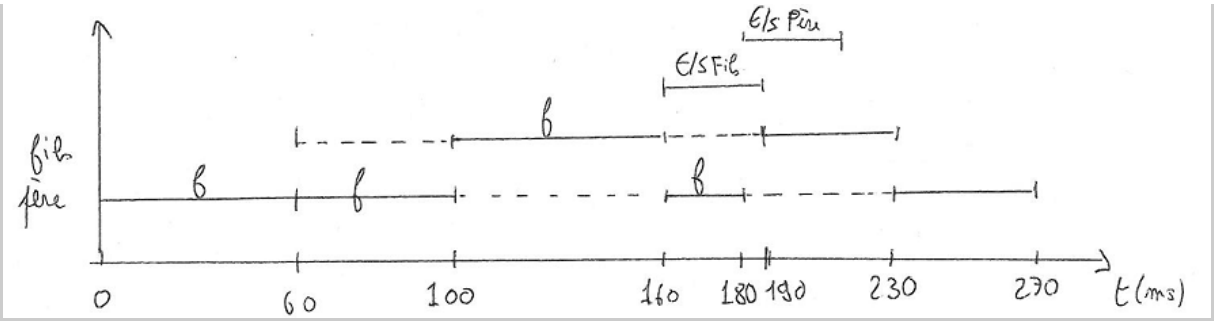


Terminal : <pidFils> 0

Taux : 260/260

I.2. (2,5 points)

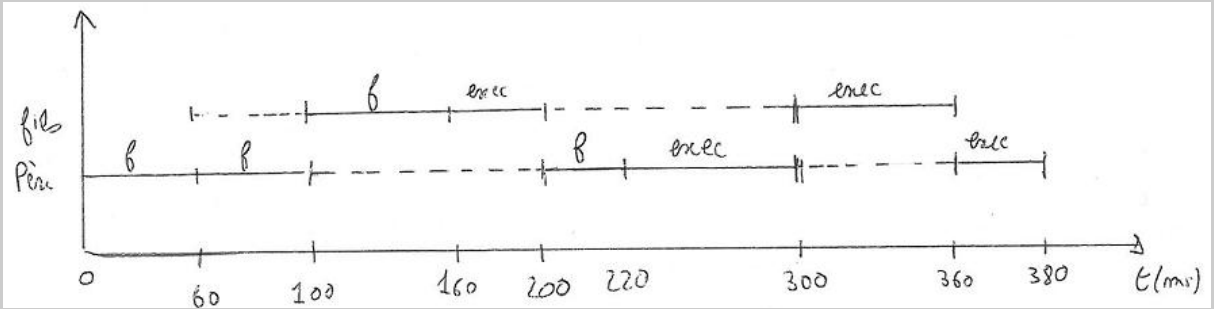
Même question pour un mode "temps partagé" avec un quantum de 100ms



Terminal : 0 <pidFils>
Taux : 260/270

I.3. (2,5 points)

On remplace maintenant la fin du code par un exec d'un programme "monProg" s'exécutant pendant 100ms sans E/S.
Donnez le diagramme de Gantt (et uniquement le diagramme) dans le cas du mode "temps partagé" avec un quantum de 100ms

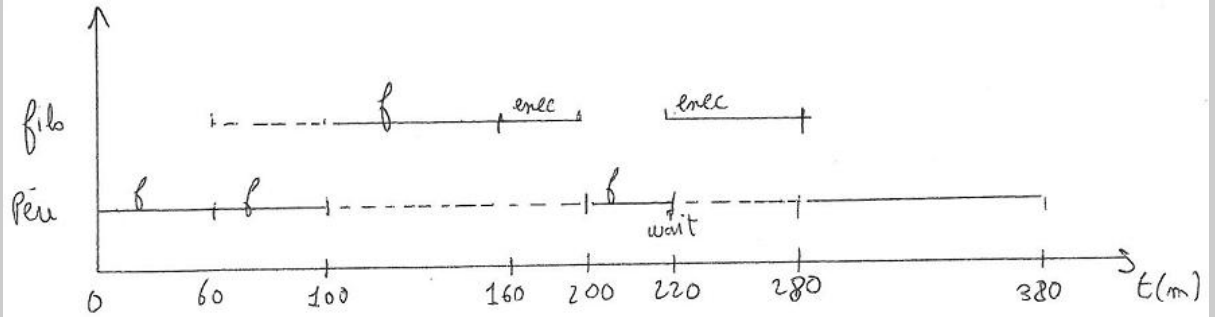



```
int main(int argc, char** argv) {
    f();
    int p=fork();
    f();
    execv("./monProg", "./monProg", null);
    return (EXIT_SUCCESS);
}
```

I.4. (2,5 points)

On ajoute pour finir un wait sachant qu'un wait est non bloquant pour un processus qui n'a jamais eu de fils.
Donnez le diagramme de Gantt (et uniquement le diagramme) dans le cas du mode "temps partagé" avec un quantum de 100ms.

```
int main(int argc, char** argv) {
    f();
    int p=fork();
    f();
    wait(NULL);
    execv("./monProg", "./monProg", null);
    return (EXIT_SUCCESS);
}
```



I.5. (1 point)

En vous basant sur vos 4 diagrammes précédents, indiquez dans quel(s) cas il y a l'apparition d'un zombie. Vous donnerez alors le temps ou le processus demeure dans cet état.

Cas 2 : processus fils est zombi pendant 40 ms
Cas 3 : processus fils est zombi pendant 20 ms

II. SYNCHRONISATION (9 POINTS)

Nous considérons 1 processus coordinateur et N processus de calcul. Les processus de calcul manipulent une variable partagée *a*. Ils sont cycliques et exécutent une boucle infinie. Chaque processus de calcul a un identifiant « id » qui varie de 1 à N : le premier processus a id =1, le second id = 2, …, le dernier id = N.

Le coordinateur initialise la variable *a*, puis réveille les processus de calcul et attend la fin de leur traitement.

Le pseudo code des processus est le suivant :

```
/* Sémaphores */
1 : Sem Travail = CS(0);      /*  bloque les processus de calcul*/
2 : Sem Mutex = CS (1) ;      /* protège la variable partagée*/
3 : Sem Fin = CS (0) ;        /* bloque le coordinateur  */

/* Variables Partagées */
4 : int a=0;

/* Processus coordinateur */
5 : void Coordinateur(){
6 :     a = 1;
        /* Débloquer les processus de calcul*/
7 :     for (i=0 ; i< N ; i++)
8 :             V(Travail) ;
        /* Attendre la fin des calculs */
9 :     for (i=0 ; i< N ; i++)
10:             P(Fin) ;
11 :    printf("valeur finale a = %d\n", a);
12 :}

/* Processus de calcul */
```

```
13 :void calcul(int id){
        /* Boucle infinie */
14:     while (1) {
                /* Attendre */
15 :            P(Travail) ;
                /* Calcul */
16 :            P(Mutex) ;
17 :            a = a *id ;
18:             V(Mutex) ;
        /* Réveil coordinateur */
19 :            V(Fin)
20 :    }
21 :}
```

II.1. (2 points)

On observe dans certains cas que le coordinateur obtient une valeur de *a* égale à 1 à la fin de son exécution et donc affiche.
« valeur finale a = 1 »

a) Expliquer dans quel cas cela est possible ?
b) Est-il possible que $a = N^N$ (N puissance N) ?
c) Est-il possible que $a = N^{N+1}$?
Justifiez vos réponses

a= 1, il suffit que le processus de calcul avec id = 1, boucle N fois
$a = N^N$, il suffit que le processus de calcul avec id =N, boucle N fois
$a = N^{N+1}$ impossible car le processus N ne peut boucler N+1 fois

II.2. (2,5 points)

On souhaite maintenant que *a* soit systématiquement égal à *N !* lorsque le coordinateur se termine (*a = 1*2*3*…*N*).
Modifiez les synchronisations en conséquence en utilisant un **tableau de sémaphores.** Vous indiquerez clairement les valeurs initiales des sémaphores ajoutés.

Il suffit de remplacer Travail par Travail[N]
Dans la boucle de réveil remplacer V(Travail) par V(Travail[i]).
Dans les processus de calcul remplacer P(Travail) par P(Travail[id-1])

II.3. (4,5 points)

On a ajoute une étape à la fin des processus de calcul.
Une fois que **tous ont terminé** le premier calcul (a = a * id), chaque processus de calcul doit afficher son id.
Ensuite, le processus coordinateur doit afficher la valeur de *a*.

Les id doivent être **affichés dans l'ordre 1, 2 … , N** : le processus avec id 1 doit faire l'affichage en premier, puis celui dont id =2 et ainsi de suite jusqu'au processus avec id = N.
Modifiez votre programme pour ajouter cette étape.

```
/* Sémaphores */
Sem Travail2[N+1] ;
for (i=0 ; i<N+1; i++)
        Travail2 [i]=CS(0) ;

5 : void Coordinateur(){
6 :     a = 1;
/* Débloquer les processus de calcul*/
7 :     for (i=0 ; i< N ; i++) {
8 :             V(Travail[i]) ;
        }
/* Attendre la fin des calculs */
9 :     for (i=0 ; i< N ; i++)
10:             P(Fin) ;
11:     V(Travail2[0]) ;
12 :    P(Travail[N]) ;
13 :    printf("valeur finale a = %d\n", a);
14 :}

/* Processus de calcul */
14 :void calcul(int id){
/* Boucle infinie */
15:     while (1) {
                /* Attendre */
16:             P(Travail[id-1]) ;
                /* Calcul */
17:             P(Mutex) ;
18:             a = a *id ;
19:             V(Mutex) ;
        /* Réveil coordinateur */
20:             V(Fin)
21:             P(Travail2[id-1]) ;
22:             printf(« id = %d\n », id) ;
23:             V(Travail2[id]);
24:     }
25 :}
```