Programmation Avancées sous POSIX	Durée : 2 heures
2010-2011	Toute documentation autorisée
Partiel	Barème indicatif

1. PROCESSUS

Considérez le programme ci-dessous. Pour simplifier on considère que l'appel à la fonction *fork ()* réussit toujours.

```
1: int main (int argc, char * argv[]) {
  2:  int i=0; int N=4;
  3:  for (i=0; i < N; i++)
  4:      if (fork () ==0) {
  5:        printf ("i:%d \n", i);
  6:        if (i%2 == 0) {
  7:          while ( (fork () == 0 ) && (i++ <N))
  8:                printf ("i:%d \n", i);
  9:          exit (0);
  10:     }
  11:     else
  12:       exit (0);
  13:  }
  14:}
```

1.1 (3 pts)

Donnez l'arborescence de processus créés ainsi que leur affichage.

Nous voulons remplacer l'appel au *printf* de la ligne 8 du programme par l'appel à *exec* de la commande *echo* qui se trouve sous le répertoire /bin.

1.2 (2 pts)

Donnez les instructions pour un tel remplacement. Donnez aussi l'arborescence de processus créés et leur affichage.

1.3 (2 pts)

Considérez le programme original. Changez-le pour que le processus principal ne se termine qu'après que tous les autres se soient terminés.

2. IPC SYSTEM V ET SIGNAUX

Considérez un programme qui crée *N* processus fils où chaque processus fils doit afficher son propre *pid* et se terminer. Cependant, un processus fils ne peut se terminer qu'après que tous les processus aient affiché leur *pid*. (Idée : stocker les pid dans un segment de mémoire partagée)

2.1 (3 pts)

Donnez le code de ce programme en utilisant des sémaphores système V pour synchroniser la terminaison des processus avec l'affichage décrite ci-dessus.

2.2 (5 pts)

Nous voulons maintenant modifier le code du programme de la question précédente de façon que le $i^{ème}$ processus fils créé (2<=i <=N) ne se termine qu'après avoir reçu un signal SIGUSR1 du $i^{ème\ -1}$ fils, c'est-à-dire le fils qui le précède lors de sa création.

3. THREAD

Nous voulons créer une chaîne de N threads **détachées** : la thread main crée une thread détachée qui crée elle aussi une thread détachée, etc. jusqu'à avoir une chaîne de N threads détachées. Après avoir affiché son propre *tid*, une thread doit se terminer. Cependant, la thread main ne peut se terminer qu'après que toutes les autres aient terminés.

3.1 (5 pts)

Donnez le code du programme ci-dessus en utilisant des **variables de condition** pour contrôler la terminaison des threads. L'utilisation de sémaphores est prohibée.