Consolidation de Février 2019 (1h)

RS : Réseaux et Systèmes

Deuxième année

★ Exercice 1: Questions de cours (5 pts)

▷ **Question 1** (1 point)**:** Expliquez précisément ce que fait l'appel système `fork`.

▷ **Question 2** (1 point)**:** Expliquez précisément ce que fait l'appel système `exec`.

▷ **Question 3** (2 point)**:** Qu'est-ce qu'un interblocage ? Comment prévenir les interblocages ? Donnez et expliquez trois méthodes.

▷ **Question 4** (1 point)**:** Lors du TP3, un étudiant a écrit le Programme 3 ci-dessous. Que pouvez-vous dire des différents affichages de `getpid()` ? Pourquoi ?

★ Exercice 2: Clonage de processus (4 pts)

▷ **Question 1:** Dessiner (comme lors du TD 1) l'exécution du Programme 1 ci-dessous. Combien de "hello !" affiche-t-il ?

▷ **Question 2:** Dessiner (comme lors du TD 1) l'exécution du Programme 2 ci-dessous (y compris ses affichages – pour les PID, vous pouvez supposer que le premier processus a le PID 1000, par exemple).

Programme 1

```
1  void doit() {
2    fork();
3    fork();
4    printf("hello!\n");
5  }
6  int main() {
7    doit();
8    printf("hello!\n");
9    exit(0);
10 }
```

Programme 2

```
1  int main() {
2    int n = 3;
3    int i;
4    printf("proc %d fils de %d\n",
5           getpid(),getppid());
6    for (i=0; i<n; i++) {
7      if (fork() == 0) {
8        printf("proc %d fils de %d\n",
9               getpid(),getppid());
10     } else {
11       wait(NULL);
12       exit(0);
13     }
14   }
15 }
```

Programme 3

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <pthread.h>
4  #include <sys/types.h>
5  #include <unistd.h>
6
7  #define NBTH 10
8
9  int somme = 0;
10
11 void * ajouteur(void * arg) {
12   int numthread = * (int*) arg;
13   int i;
14   printf("ajouteur: getpid()=%d\n",
15          getpid());
16   for (i = 0; i < 1000000; i++) {
17     somme += numthread;
18   }
19   return NULL;
20 }
21
22 int main(int argc, char** argv) {
23   int i;
24   pthread_t threads[NBTH];
25   printf("main: getpid()=%d\n",
26          getpid());
27   for (i = 0; i < NBTH; i++) {
28     pthread_create(
29        &threads[i], NULL,
30              ajouteur, (void*) &i);
31   }
32   for (i = 0; i < NBTH; i++) {
33     pthread_join(threads[i], NULL);
34   }
35   printf("somme=%d\n", somme);
36   return 0;
37 }
```

★ Exercice 3: Utiliser `fork`, `exec`, `wait`, `waitpid`, `pipe`, `dup`, `dup2` (6 points)

Donner le code C (sans utiliser la fonction `system`) correspondant à ce que fait un *shell* lorsqu'on tape les lignes de commandes suivantes. Afin d'obtenir un code lisible et court, vous êtes dispensés des tests d'erreur des appels systèmes.

▷ **Question 1:** `prog1 < fichier1.txt`
(Il faut donc lancer le programme *prog1* après avoir redirigé son entrée standard pour qu'il reçoive le contenu du fichier au lieu de ce qui est écrit au clavier)

▷ **Question 2:** `prog1 && prog2`
(Il faut donc : lancer un programme *prog1*, attendre sa fin, récupérer son code de retour, et si ce code de retour est égal à 0, lancer un programme *prog2* et attendre sa fin)

▷ **Question 3:** `prog1 | prog2`
(Il faut donc : lancer deux programmes *prog1* et *prog2*, en liant la sortie standard de *prog1* à l'entrée de *prog2*)

Quelques fonctions utiles

```
pid_t fork(void);
int execlp(const char *file, const char *arg, ...);
pid_t wait(int *status);
pid_t waitpid(pid_t pid, int *status, int options);
WIFEXITED(status) -- returns true if the child terminated normally
WEXITSTATUS(status) -- returns the exit status of the child
int pipe(int pipefd[2]);
int dup(int oldfd);
int dup2(int oldfd, int newfd);
```

★ Exercice 4: Savoir utiliser les sémaphores et reconnaître les schémas de synchronisation classiques (5pts)

Suite à une expérimentation malheureuse, un début d'incendie s'est déclaré au club robotique. Immédiatement, tous les étudiants présents sur place et au foyer se précipitent à la cafet pour remplir des carafes d'eau avant de les jeter sur les flammes. Un seul robinet, des couloirs exigus et des dizaines d'étudiants courant en tout sens pour porter de l'eau vers les flammes : le chaos est indescriptible.

▷ **Question 1:** Proposez une solution à base de sémaphore(s) pour d'éviter que les étudiants se bousculent devant le point d'eau. De quel schéma de synchronisation classique cela se rapproche-t-il ? Pourquoi ?

▷ **Question 2:** Les porteurs d'eau perdent un temps précieux à contourner le bar et attendre leur tour au robinet. Proposez une solution à base de sémaphore(s) où l'un d'entre eux seulement est en charge de remplir les carafes pour les autres, qui courent ensuite en jeter le contenu dans les flammes quand elles sont pleines. De quel schéma de synchronisation classique cela se rapproche-t-il ? Pourquoi ?