Master1 – POSIX	- Durée : 2 heures
2006	- Toute documentation autorisée
Contrôle Continu	- Barème indicatif
	Luciana Arantes, Bertil Folliot

1. EXERCICES (4 POINTS)

a) Ecrivez un programme qui crée un processus "zombie" pendant 10 secondes. (1,5 pt)

b) Ecrivez une fonction "handler" de signal qui affiche le PID et la valeur de retour de tous les fils terminés (plusieurs signaux SIGCHLD peuvent avoir été émit). (2,5 pts)

2. GESTIONS DES PROCESSUS (10 POINTS)

Considérez le programme ci-dessous qui crée une chaîne de N-1 processus fils.

```
1:    #define N 4
2:    pid_t pid_fils;
3:    int main (void) {
4:       int i=0;
5:       while (i<N-1) {
6:           if ((pid_fils=fork()) == -1 )
7:             exit (1);
8:          if (pid_fils != 0)  {
9:             if (i!=0)
10:               printf ("i:%d-%d,%d\n",i, pid_fils, getppid());
11:            break;
12:         }
13:         i++;
14:       } /*while */

15:       if (i!=N-1)
16:         wait (NULL);
17:      return 0;
18:   }
```

2.1 (3 pts)

Détaillez ce que fait ce programme.

2.2 (3 pts)

Supposons que nous **enlevons les lignes 15 et 16** du programme et que vous ne pouvez utiliser, ni la fonction ***wait,*** ni le signal ***SIGCHLD*** dans votre programme. Cependant, nous voulons qu'un processus ne se termine qu'après que tous les autres aient été créés. Modifiez le programme en conséquence en utilisant des signaux (sauf *SIGCHLD*).

Reprenons le programme original. Nous pouvons le voir comme un anneau de processus. Le voisin de droite du processus *i* est le processus *i+1* (fils de *i*), sauf pour le processus N-1,

dont le voisin de droite est le processus 0. Le voisin de gauche du processus *i* est le processus *i-1* (père de *i*), sauf pour le processus 0, dont le voisin de gauche est le processus N-1.

2.3 (4 pts)

Modifiez le programme pour que les processus 0 et N-1 affichent aussi les identifiants (les PID) de leurs voisins de gauche et de droite. Indiquez quel mécanisme vous utilisez.

3. FICHIERS (6 POINTS)

Considérez le programme ci-dessous dans lequel deux processus (père et fils) écrivent dans le fichier "./fich1".

```
1:int main (void) {
2:   int fd1, fd2, fd3;

3:   if ((fd1 = open ("./fich1", O_RDWR| O_CREAT |
4:    O_SYNC,0600)) == -1)
5:      return EXIT_FAILURE;
6:   if (write (fd1,"abcde", strlen ("abcde")) == -1)
7:      return EXIT_FAILURE;

8:   if (fork () == 0) {
9:      if ((fd2 = open ("./fich1", O_RDWR)) == -1)
10:         return EXIT_FAILURE;
11:     if (write (fd1,"123", strlen ("123")) == -1)
12:         return EXIT_FAILURE;
13:     if (write (fd2,"45", strlen ("45")) == -1)
14:         return EXIT_FAILURE;
15:  }
16:  else {
17:       fd3=dup(fd1);
18:       if (lseek (fd3,0,SEEK_SET) == -1)
19:        return EXIT_FAILURE;
20:         if (write (fd3,"fg", strlen ("fg")) == -1)
21:           return EXIT_FAILURE;
22:       if (write (fd1,"hi", strlen ("45")) == -1)
23:           return EXIT_FAILURE;
24:       wait (NULL);
25:       close (fd1); close(fd2); close(fd3);
26:  }
27:  return EXIT_SUCCESS;
28:}
```

3.1 (3 pts)

Quelles sont les contenus possibles du fichier "./fich1 " ?

3.2 (3 pts)

En utilisant des signaux, changez le programme pour que la taille du fichier soit toujours égale à 8.