Test de Connaissances de Systèmes d'Exploitation 2009

1 Questions

1. Qu'est-ce qu'un zombie ?
2. Qu'est-ce qu'un démon ?
3. Que fait le processus `init` au sujet des zombies, et dans quelles circonstances ?
4. Que va afficher le programme suivant, en supposant que tous les fork() réussissent ?

```
#include <iostream>
#include <unistd.h>

using namespace std;

int main() {
  for(int kProc(1);kProc < 3;kProc++) {
     cout << kProc << "\n";
     if(fork() > 0) {
        cout << (10 * kProc)  << "\n";
     }
     else {
        cout << (100 * kProc) << "\n";
     }
  }
  return(0);
}
```

2 Exercice

1.1 Écrivez un programme qui crée trois processus apparentés ainsi : père, fils et petit-fils.

1.2 Modifiez le programme précédent de sorte que chaque père attende son fils, et écrive sa valeur de retour s'il est sorti normalement.

2.1 Écrivez un programme qui crée vingt processus apparentés ainsi : un père, le reste des fils.

2.1 Rajoutez au programme ce qu'il faut pour qu'on ne crée pas de zombies même si le père n'appelle ni `wait()` ni `waitpid()`.

3 Exercice

1. Faites un programme qui déclare un pipe anonyme, se duplique, dans le processus père écrit un entier (codé en représentation machine), et dans le processus fils il le lit et l'affiche à l'écran.
2. Considérons un programme qui déclare un pipe nommé, et qui prend un argument sur la ligne de commande représentant un message. Le programme écrit ensuite ce message sur le pipe nommé. Utilisez pour cette écriture un protocole permettant à un autre programme de bien lire ce message depuis le pipe nommé, sachant que le message peut contenir un nombre arbitraire de caractères alphanumériques (i.e. lettres majuscules, lettres minuscules, chiffres).

4 Exercice

1. Soient trois fichiers `fic1.txt`, `fic2.txt` et `fic3.txt`. Écrivez trois programmes p_1, p_2 et respectivement p_3, se comportant comme suit. Le processus p_1 lit des nombres entiers non-négatifs depuis le fichier `fic1.txt`, codés en représentation machine. Le processus p_2 lit des lettres de l'alphabet depuis le fichier `fic2.txt`. Le processus p_3 reçoit toutes ces informations (autant de lettres que de nombres) depuis p_1 et p_2. Toujours le processus p_3 écrit dans un fichier `fic3.txt` les lettres reçues depuis p_2, en en répétant chacune autant de fois que le dit le nombre correspondant lu depuis p_1. Voici un exemple de déroulement possible, étape par étape :

`fic1.txt`	p_1	p_3	p_2	`fic2.txt`	`fic3.txt`
$2, 3, 1$	$\rightarrow$lit $2, 3, 1$		lit `abc`$\leftarrow$	`abc`	
	écrit $2, 3, 1 \rightarrow$		$\leftarrow$écrit `abc`		
		$\rightarrow$lit $2, 3, 1$			
		lit `abc`$\leftarrow$			
		écrit `aabbbc` $\longrightarrow$			`aabbbc`

   Votre programme doit fonctionner avec des tailles arbitraires des fichiers `fic1.txt` et `fic2.txt`.

2.1 Le processus p_3 écrit sur un pipe nommé (« fifo »), à l'intention de p_1 et de p_2, un entier (codé en représentation machine) signifiant la confirmation de la réception de l'information. Chaque confirmation sera écrite deux fois, une fois pour le processus p_1, et une fois pour le processus p_2.

2.2 Rédigez la partie lecture correspondante, qui sera mise par la suite dans p_1 et p_2 (elle est la même).

2.3 En utilisant $P()$ et $V()$, marquez les endroits du programme du processus p_1 ainsi que les endroits du programme du processus p_2 où on devrait se servir d'un sémaphore pour que les processus p_1 et p_2 lisent depuis le pipe nommé les confirmations à tours de rôle.

2.3 Écrivez les instances de $P()$ et de $V()$ en détail utilisant les fonctions pour les sémaphores.