L3

Contrôle de Programmation Unix – 8 novembre 2016

Michel SOTO

AUCUN DOCUMENT AUTORISÉ

Durée: 1 H 30

Le barème est indicatif - Nombre de pages: 2

La concision de vos réponses et la propreté de votre copie seront prises en compte.

PARTIE I : CONNAISSANCE DU COURS

Question 1 (3 points)

Répondez aux affirmations suivantes uniquement par "VRAI", ou "FAUX" ou "NE SAIS PAS".

Barème : **réponse exacte : +1 point, réponse fausse : -0,5 point sur la copie, "ne sais pas" : 0 point**

a) Si le père d'un processus est terminé et que ce processus fait appel à la primitive `getppid` alors cette dernière retourne la valeur –1.

b) Tout signal peut être capté.

c) La primitive `signal` est utilisée pour envoyer un signal à un processus.

Question 2 (2 points)

a) Un processus fait un appel à la primitive open(). Quelles sont les tables du système qui sont concernées par cet appel.

b) Citez 2 façons dont un processus zombie peut disparaître du système.

PARTIE II : APPLICATION DU COURS

Question 3 (3 points)

Le programme suivant compile et s'exécute parfaitement. Pour chaque valeur de la variable `i`, donnez le résultat affiché lors de son exécution. Justifiez votre réponse.

```
main(){
      int i;
      for (i=1; i <= 2; i++){
            fork();
            printf ("%d – Hello !!\n", i);
      }
} // main
```

* * * * *

Dans les questions suivantes, les programmes devront être rédigés selon les règles de l'art : vérification du nombre de paramètres éventuels, vérification des valeurs de retour des primitives du système, indentation et propreté du code. Vous êtes dispensés des `includes`.

Question 4 (6 points)

Ecrire un programme `auto_exec.c` qui se recouvre N fois lui-même. A chaque recouvrement le programme affiche : `./auto_exec : recouvrement i`.

Avant de se terminer, il affiche : `./auto_exec : terminé`

Par exemple : `./auto_exec 3` aura pour résultat:

```
./auto_exec : recouvrement 1.
./auto_exec : recouvrement 2.
./auto_exec : recouvrement 3.
./auto_exec : terminé
```

Question 5 (6 points)

Ecrire un programme `horloge.c` dont l'exécution crée 3 processus H, M, S qui gèrent respectivement les compteurs des heures, des minutes et des secondes d'une horloge numérique. La synchronisation des 3 processus se fera exclusivement au moyen des signaux et de `sigaction`.

Chaque processus n'affiche la valeur de son compteur que lorsque sa valeur change.

Lancement du programme: `./horloge h m s` où `h`, `m` et `s` sont respectivement la valeur initiale des heures, des minutes et des secondes de l'horloge.

ANNEXE

```
NAME
       execl, execlp, execle, execv, execvp
SYNOPSIS
       int execl(const char *path, const char *arg, ...);
       int execlp(const char *file, const char *arg, ...);
       int execle(const char *path, const char *arg,
                  ..., char * const envp[]);
       int execv(const char *path, char *const argv[]);
       int execvp(const char *file, char *const argv[]);

NOM
       sigaction
SYNOPSIS
      int sigaction(int signum, const struct sigaction *act,
                    struct sigaction *oldact);

      struct sigaction {
            void (*sa_handler)();
            sigset_t sa_mask;
            int sa_flags;
      }
```