L3

Contrôle de Programmation Unix – 20 octobre 2015

Michel SOTO

AUCUN DOCUMENT AUTORISÉ

Durée: 1 H 30

Le barème est indicatif - Nombre de pages: 2 sur une feuille

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

PARTIE I : CONNAISSANCE DU COURS

Question 1 (4 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) Tous les threads se terminent lorsque le processus auquel ils appartiennent se termine.

b) Soit un processus F fils d'un processus P. Si P ou F s'apprête à modifier une de ses variables, le mécanisme de `copy-on-write` effectue, avant la modification, une copie intégrale de P (moins les caractéristiques qui ne sont jamais héritées).

c) La primitive `_exit` exécute les gestionnaires de fin installés avec `atexit`.

d) La table des `i-nodes` est une copie en mémoire centrale de la `i-list` située sur le disque

Question 2 (4 points)

a) A quoi sert le bit `set-uid` (bit `u`) du champ `st_mode` lorsqu'il est positionné sur un fichier contenant un exécutable ? Citez une commande du système qui possède le bit `set-uid` positionné sur son exécutable. Précisez à quoi sert cette commande.

b) Qu'est-ce qu'un thread joignable ?

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

PARTIE II : APPLICATION DU COURS

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 3 (3 points)

Ecrire un programme qui se recouvre N fois lui-même. A chaque recouvrement le programme affiche :

```
son_nom : recouvrement n°i.
```

Avant de se terminer, il affiche :

```
son_nom terminé.
```

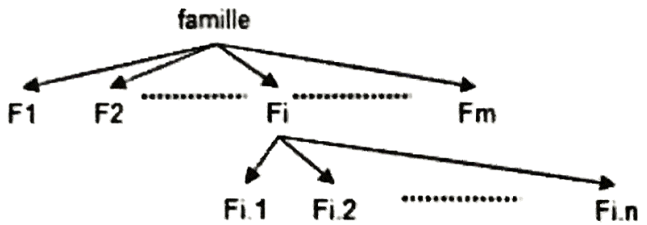
Question 4 (4 points)

Ecrire un programme `make_env` qui reçoit en paramètres le nom d'un **quelconque** programme à exécuter et une liste de valeurs de la forme `NOM1=VALEUR1`, `NOM2=VALEUR2`, etc. Lors de son exécution, `make_env` construit un environnement pour le programme à exécuter avec la liste des valeurs qu'il a reçues et lance le programme avec l'environnement qu'il a construit. Si le lancement échoue, `make_env` affiche un message :

```
Erreur: nom_du_programme
```

Question 5 (5 points)

Ecrire un programme `famille` auquel 3 entiers `m` (≥0), `n` (≥0) et `i` (∈ [0 ; m]) sont passés, dans cet ordre, au moment de son exécution afin qu'il construise une famille de processus respectant le modèle suivant :



- `famille` affiche son nom (`famille`) et son `PID`
- `famille` se termine si et seulement si tous ses petits-fils sont effectivement terminés. Il affiche alors :

```
FIN FAMILLE
```

- Chaque fils et petit-fils affiche son nom (`Fi` ou `Fi.j`), son `PID`, le `PID` de son père et de son grand-père.

ANNEXE

NAME

execl, execlp, execle, execv, execvp - execute a file

SYNOPSIS

```
#include <unistd.h>

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[]);
```