L3

Contrôle de Programmation Unix – 4 novembre 2014

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) Un processus zombie ne devient pas orphelin quand sont père se termine.

b) Le système gère une table `file table` par processus.

c) Par défaut, les fichiers ouverts sont fermés après l'appel à une primitive exec.

d) Un `fork` ne provoque pas la copie des données du processus père.

Question 2 (2 points)

Complétez le programme suivant afin qu'il affiche le contenu de sa variable d'environnement `PATH`.

```
main (int argc, char *argv[], char **arge){
...
} // main
```

Question 3 (3 points)

a) Qu'est-ce que qu'une i-list et à quoi sert-elle ?

b) Le shell a-t-il un statut particulier du point de vue du système ? Justifiez votre réponse

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 système et indentation du code. Vous êtes dispensé des `includes`.

Question 4 (4 points)

Ecrire un programme qui crée 1 à N (N est passé en paramètre de la fonction `main`) processus tels que P_N est le fils de P_{N-1} qui le fils de P_{N-2}, etc. Chaque P_i s'endort i secondes puis attend la fin de son fils P_{i+1} sauf P_N qui ne fait que s'endormir N secondes

Question 5 (7 points)

Ci-dessous, le code du mini shell vu en TD.

a) Ce mini shell se duplique parfois de façon indésirable comme le montre le résultat de la commande `ps`.

```
mishell> ps l
F   UID   PID  PPID PRI  NI    VSZ   RSS WCHAN  STAT TTY        TIME COMMAND
0  2008 17041 17039  20   0 114968  2116 wait   Ss   pts/2      0:00 -bash
0  2008 17069 17041  20   0   4136   280 wait   S+   pts/2      0:00 ./minishell
1  2008 17090 17069  20   0   4136   332 wait   S+   pts/2      0:00 ./minishell
0  2008 17628 17090  20   0 105984   768 -      R+   pts/2      0:00 ps l
```

1. Dans quelle situation cette duplication se produit-elle ? Justifiez votre réponse.
2. Quelle correction faut-il apporter au code afin que cette duplication ne se produise plus ?

b) Ce mini shell produit aussi de manière indésirable des processus à l'état zombi qui occupent **inutilement** la table des processus du système.

1. Dans quelle situation ces processus zombis sont-ils produits ? Justifiez votre réponse.
2. En utilisant les seules notions présentées en cours, présentez en quelques lignes une solution afin que disparaisse systématiquement ces zombis inutiles.
3. Ecrivez le code de la solution que vous proposez en indiquant où il se situe dans le code ci-dessous.

```
#include <stdio.h>
#include <string.h>
#include <signal.h>
#include <unistd.h>

#define TAILLE_LIGNE 250
#define TAILLE_ARG    40
#define EVER (;;)

char commande [TAILLE_LIGNE];

/*=================================================*/
  main(){
/*=================================================*/
char *arg[TAILLE_ARG];
char *bg, char *mot;
int i, pid, CodeRetour;

/*1*/for EVER {// Boucle infinie

/*2*/    printf ("mishell> ");

     // Lecture de la ligne de commande
     // entrée par l'utilsateur
/*3*/    fgets(commande, TAILLE_LIGNE, stdin);

     // Suppression du retour chariot
/*4*/    commande[strlen(commande)-1]='\0';

     // Recherche et suppression du & eventuel
/*5*/    if (bg=strrchr(commande, '&')) *bg='\0';

    // Recherche des mots separés par un blanc
/*6*/  for(i=0,mot=strtok(commande, " ");
           mot!=NULL;
             mot=strtok(NULL," "),i++){

     // Preparation des parametres du execvp
/*7*/   arg[i]=(char *)malloc(strlen(mot)+1);
/*8*/   strcpy(arg[i],mot);

/*9*/  }//for

/*10*/ arg[i]=NULL;

    // La ligne de commande est-elle vide ?
/*11*/ if (i>0)
/*12*/     if (fork()==0) {// CODE DU FILS
/*13*/         execvp (arg[0], arg);
/*14*/         perror ("execvp");
/*15*/      }

/*16*/      else {// CODE DU PERE
/*17*/            if (bg!=NULL)
/*18*/               // Pas d'attente
/*19*/            else {// Attente
/*20*/                  pid=wait(&CodeRetour);
/*21*/            }// if (bg!=NULL)
/*22*/      }// if (fork()==0)

/*23*/    }// for EVER

/*24*/ }// main()
```

ANNEXE

```
NOM
       strstr - Rechercher une sous-chaîne.

SYNOPSIS
       #include <string.h>
       char *strstr(const char *meule_de_foin, const char *aiguille);

DESCRIPTION
       La   fonction  strstr()  cherche  la  première  occurrence  de  la  sous-chaîne
aiguille au sein de la chaîne meule_de_foin.

VALEUR RENVOYEE
       Cette fonction renvoie un pointeur sur le début de la  sous-chaîne,  ou  NULL
si celle-ci n'est pas trouvée.
```