Systèmes - M1 - Juin 2011

Exercice 1

Écrire en C le code d'une commande de nom `protection` prenant en argument :

1. une suite, éventuellement de longueur nulle, d'arguments `-s n`
2. suivie de l'argument `-e` lui-même suivi d'un nom d'exécutable suivi de ses propres arguments

Exemple : `protection -s 3 -s 9 ls /usr/bin`

La commande `protection` aura pour effet d'exécuter la commande spécifiée avec ses arguments en protégeant cette exécution contre les signaux indiqués (`-s n` pour le signal de numéro `n`).

Lorsque la *sous*-commande se termine, la commande protection doit se terminer et sa valeur de retour devra être :

1. si la *sous*-commande s'est terminée normalement
2. si la *sous*-commande s'est terminée par réception d'un signal

Exercice 2

On suppose avoir à sa disposition un exécutable `affiche` recevant deux arguments et capable d'afficher tous les nombres compris entre les nombres reçus en argument (exemple : `affiche 0 49`).

Écrire en C, une commande permettant de lancer deux processus exécutant chacun une instance du programme `affiche` de sorte qu'à l'écran l'affichage en ordre des nombres de 0 à 99 soit réalisé.

Exercice 3

Soit le programme suivant :

```
#include <sys/types.h>
#include <signal.h>
#include <unistd.h>
int ok;
void capture(int sig) {
        wait(NULL); ok = 1;
        printf("Tout va bien...\n");
}
int main(int argc,char *argv[]) {
        struct sigaction action;
        sigset_t ens;
        ok = 0;
        action.sa_handler = capture;
        action.sa_flags = 0;
        sigemptyset(&(action.sa_mask));
        sigaction(SIGUSR1,&action,NULL);
        if (fork()==0) {
                printf("Je fais quelque chose\n");
                kill(getppid(),SIGUSR1);
                exit(0);
        }
        if (ok==0) { pause(); }
        printf("Moi aussi je bosse\n");
```

```
}
```

1) Que tente de faire ce programme ? Que souhaite t'on afficher ?
2) Que se passe t-il si le fils effectue l'appel à `kill()` avant que le père n'ait eu le temps de faire appel à `pause()` ?
3) Corriger pour que cela fonctionne dans tous les cas.

Exercice 4

1) Écrire un programme dont l'exécution conduit à la création d'un processus orphelin dont le fils est zombi. L'exécution doit permettre de les observer tous les deux depuis un shell.
2) Modifier le programme de sorte que l'envoi, au processus orphelin, du signal `SIGUSR1` conduise à la disparition des deux. Expliquer comment cela se produit exactement.

Exercice 5

1) Pourquoi les *handler*s de capture de signaux sont conservés lors d'un appel à `fork()` mais pas hérités lors d'un appel à `exec()` ?
2) Pourquoi est-il utile de pouvoir bloquer temporairement la délivrance de signaux ? Donner un exemple significatif.
3) Pourquoi est-il conseillé de faire en sorte que le *handler* positionné à réception du signal `SIGSEGV` interrompe l'exécution du processus ?

Problème

On vous propose d'écrire un système de réservation. Les utilisateurs de ce système sont identifiés par leur uid, les objets à réserver sont identifiés par un entier (positif). Il ne doit pas être possible à un utilisateur de réserver plus d'un objet, et à un objet d'être réservé par deux utilisateurs.

La base de données prendra la forme de deux fichiers. Le premier (`objets`) contient des entiers identificateurs d'utilisateurs (`0` pour indiquer qu'un objet est libre), et le second (`utilisateurs`) contient des entiers identificateurs d'objets (`0` pour indiquer qu'un utilisateur n'a rien réservé). Ex : si l'utilisateur `33` a réservé l'objet `12` alors :

1 le `33`e int du fichier `utilisateurs` contiendra la valeur `12`,
2 le `12`e int du fichier `objets` contiendra la valeur `33`.

3. écrire une commande `libre` permettant d'obtenir la liste des objets actuellement libres.
4. écrire une commande `annule` permettant à l'utilisateur d'annuler sa réservation.
5. écrire une commande `reserve` *`numéro`* permettant à l'utilisateur de réserver l'objet de numéro spécifié, s'il n'a encore rien réservé.
6. comment empêcher un utilisateur d'écrire un programme permettant d'annuler ou créer une réservation pour un autre utilisateur que lui-même ? Indiquez quels sont les objets systèmes que vous utilisez/créez et les droits d'accès afférents.
7. l'exécution concurrente de ses diverses commandes pose-t'elle problème ? Lesquels ? Décrivez les situations dangereuses ? Comment y remédier ? Corrigez votre code.