TP2: Processus

Système D'exploitation

Samir AKNINE, Antoine GRÉA

☘ Compétences À Acquérir

- ✔ Maîtriser le cycle de vie des processus
- ✔ Savoir créer des processus en C
- ✔ Tuer des zombies

> ❗ Les questions marquées d'un drapeau ⚑ seront à rendre dans le compte-rendu global.

1 Fork Simple

Soit le programme C suivant :

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main() {
    int pid;
    printf("debut\n");
    pid = fork();
    if (pid == 0) {
         printf("execution 1\n"); }
    else {
         printf("execution 2\n"); }
    printf("Fin\n");
    return 0;
}
```

1.1 Théorie

❓ Prévoyez le résultat de l'exécution de ce programme. Donnez l'arbre des processus de ce programme.

1.2 Pratique

❓ Vérifiez votre réponse en compilant (`gcc fichierSource.c -o binaire`) puis en exécutant ce programme et en y ajoutant des affichages (par exemple, en affichant le PID du processus appelant,...).

2 Forks Imbriqués

Soit le programme suivant :

```c
int main(int argc, char *argv[]) {
    int a, e;
    a = 10;
    if (fork() == 0) {
         a = a *2;
         if (fork() == 0) {
              a = a +1;
              exit(2);
         }
         printf("%d \n", a);
         exit(1);
    }
    wait(&e);
    printf("a : %d ; e : %d \n", a, WEXITSTATUS(e));
    return(0);
}
```

2.1 Arbre Des Processus

❓ Donnez l'arbre des processus de ce programme. Indiquez ce que chaque processus affiche.

❓ On supprime l'instruction `exit(2)`, reprenez la question précédente en conséquence.

2.2 ⚑ Zombie !

Téléchargez le programme `zombie` (disponible au même endroit que le sujet). Le programme zombie grogne quand il reçoit un signal autre que `SIGBUS`. Il est également responsable de l'affichage que vous pouvez constater quand vous l'exécutez dans un terminal. Attention ne fermez pas le terminal au risque de rendre l'exercice bien plus compliquée !

❓ Créez un programme appelé `killbill` qui est capable de tuer le zombie ainsi créé. (La fonction disponible dans le fichier `parentpid.h` vous sera probablement utile)

❓ Pourquoi le zombie grogne-t-il quand il se prend le bus ? Précisez la nature de ce phénomène.

3 La Primitive `exec`

On considère le programme suivant :

```c
int main() {
    int p;
    p = fork();
    if (p == 0) {
        sleep(2);
        execl("/bin/echo", "echo", "Nonnnnnn", "!", NULL);
    }
    wait(NULL);
    printf("Non, Je suis ton père\n");
    return 0;
}
```

3.1 Théorie

❓ Donnez le résultat de l'exécution de ce programme (on suppose que l'appel `execl` est réussi). Expliquer qui affiche quoi et pourquoi.

3.2 Arbre Des Processus

Soit le programme `nemaxe` suivant en C sous UNIX, où `prog` est un programme qui ne crée pas de processus.

```
main(int argc, char*argv[]) {
    int retour;
    printf("%s\n", argv[0]);  // A
    switch (fork()) {    // B
         case -1 :
              perror("fork1()");
              exit(1);
         case 0 :
              switch (fork()) {    // C
                   case -1 :
                        perror("fork2()");
                        exit(1);
                   case 0 :
                        if (execl("./prog","prog", NULL) == -1)
                        {
                             perror("execl");
                             exit(1);
                        }
                        break;
                   default :
                        exit(0);
              }
         default :
              wait(&retour);
    }
}
```

? Représentez tous les processus créés par ce programme sous forme d'un arbre, en vous servant des lettres en commentaires. Quels sont les ordres possibles de terminaison des processus ?