Deliot Maxence
Desportes Kilian 2B

Principe des Systèmes d'Exploitation
TP3

Communication entre processus par signaux

Deliot Maxence
Desportes Kilian 2B

Table des matières

Deliot Maxence
Desportes Kilian 2B

Ex. 1 : Réception et gestion des signaux par les processus :

```
[0]utilisateur@Ubuntu:~$ ./a.out
^Cinteruption numéro 1
^Cinteruption numéro 2
^Cinteruption numéro 3
^C^C^C[0]utilisateur@Ubuntu:~$
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
int i=1;

void sighandler(){
        if(i<4){
                printf("interuption numéro %d\n",i);
                fflush(stdout);
        }else if(i==6){
                exit(0);

        }
        i++;

}

int main (){

        signal(SIGINT, sighandler);
        while(1){
                sleep(1);
        }
}
```

Ex . 2 : alarm :

```
[0]utilisateur@Ubuntu:~$ ./a.out
Compteur : 900
Compteur : 2182
Compteur : 5107
Compteur : 10249
Compteur : 14926
Compteur : 19594
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
int i=0;

void sighandler(){
        printf("Compteur : %d\n",i);
        alarm(2);
}



int main (){
        signal(SIGALRM, sighandler);
        alarm(2);
        while(1){
                usleep(1);
                i++;
        }
}
```

Ex. 3 : Détecter la fin d'un processus fils sans l'attendre :

c)

```
[0]utilisateur@Ubuntu:~$ ./a.out
Compteur : 0
Compteur : 1
Compteur : 2
Compteur : 3
Le fils s'est terminé , code de retour : 0
Compteur : 4
Compteur : 5
Compteur : 6
Compteur : 7
```

```
[0]utilisateur@Ubuntu:~$ kill 3021
```

Si on kill le processus fils dans un autre terminal, le fils se termine et le code de retour est 0.

```
[0]utilisateur@Ubuntu:~$ ./a.out
Compteur : 0
Compteur : 1
Compteur : 2
Compteur : 3
Compteur : 4
Compteur : 5
Compteur : 6
Compteur : 7
Compteur : 8
Compteur : 9
```

```
[0]utilisateur@Ubuntu:~$ kill -SIGSTOP 3030
```

Si on arrête le processus fils, le processus père va capter que son fils est en état d'attente et va s'arrêter jusqu'à ce que le fils reprenne.

```
[0]utilisateur@Ubuntu:~$ ./a.out
Compteur : 0
Compteur : 1
Compteur : 2
Compteur : 3
Le fils s'est terminé , code de retour : 0
Compteur : 4
Compteur : 5
Compteur : 6
```

```
[0]utilisateur@Ubuntu:~$ kill -SIGSTOP 3061
```

d) Si on remplace wait(&status) par waitpid(–1,&status,WNOHANG), lorsque l'on fait un SIGSTOP sur le processus fils, le processus père s'exécute toujours. Le –1 va signifier que le processus fils se met en attente et le WNOHANG signifie que le père reprend le contrôle si le fils ne se termine pas.

```c
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>

void sighandler(){
        int status=0;
        waitpid(-1, &status, WNOHANG);
        printf("Le fils s'est terminé , code de retour : %d     \n",WEXITSTATUS(status));
        fflush(stdout);
}

int main (){
        int i=0;
        pid_t rc=fork();
        if(rc<0){
                printf("Erreur fork");
                fflush(stdout);
        }else if(rc==0){
                sleep(20);
                exit(10);
        }else if(rc>0){
                signal(SIGCHLD, sighandler);
                while(1){
                        printf("Compteur : %d \n",i);
                        fflush(stdout);
                        i++;
                        sleep(2);
                }
        }
}
```

Code final de l'exercice 3.