

## Chapitre 1 – Processus :

### Exercice 1.1 :

Le but de cet exercice est de créer un programme qui lancera un processus puis s'endormira juste après. De cette façon, on essaiera de constater que le processus fils devient un processus zombie.

### Code source :

Header.h :

---

```
#ifndef HEADER_H
#define HEADER_H

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#endif // HEADER_H
```

---

Main.c :

---

```
#include "headers/header.h"

int main(int argc, char** argv) {
    printf("Programme principal\n");
    int pid = fork();

    if (pid == -1) { printf("Fork impossible\n");
    } else if (pid == 0) {
        //Le processus fils désire mourir mais doit attendre que son père
        //récupérer son état pour se terminer.
        printf("Processus fils\n");
        exit(0);
    } else {
        //Le processus père ne pourra pas récupérer l'état du processus fils
        //pour que celui-ci puisse se terminer.
        printf("Processus père\n");
        pause();
    }
    return (EXIT_SUCCESS);
}
```

---

## Trace d'exécution :

```
jeremie@Extensa-2511: ~/Developpement/NetBeansProjects/OS_Exercice_1_1
jeremie@Extensa-2511:~/Developpement/NetBeansProjects/OS_Exercice_1_1$ cd Developpement/NetBeansProjects/OS_Exercice_1_1/
jeremie@Extensa-2511:~/Developpement/NetBeansProjects/OS_Exercice_1_1$ ll
total 20K
drwxrwxrwx 3 jeremie jeremie 4,0K mai 27 22:15 build
drwxrwxrwx 3 jeremie jeremie 4,0K mai 27 22:15 dist
-rwxrwxrwx 1 jeremie jeremie 3,5K mai 24 14:36 Makefile
drwxrwxrwx 3 jeremie jeremie 4,0K mai 27 22:15 nbproject
drwxrwxrwx 3 jeremie jeremie 4,0K mai 27 22:15 src
jeremie@Extensa-2511:~/Developpement/NetBeansProjects/OS_Exercice_1_1$ ./dist/Debug/GNU-Linux/os_processus
Programme principal
Processus père
Processus fils
```

Une fois lancé, le programme crée bien deux processus, un père et un fils. En exécutant la commande « `ps -e -o pid,ppid,stat,cmd` » on obtient ceci :

```
jeremie@Extensa-2511: ~/Developpement/NetBeansProjects/OS_Exercice_1_1
1834 1086 Sl /usr/lib/gvfs/gvfsd-network --spawner :1.4 /org/gtk/gvfs/exec_spaw/2
1863 1086 Sl /usr/lib/gvfs/gvfsd-dnssd --spawner :1.4 /org/gtk/gvfs/exec_spaw/6
1909 1086 S /usr/lib/x86_64-linux-gnu/gconf/gconfd-2
4718 709 S /usr/sbin/dnsmasq --no-resolv --keep-in-foreground --no-hosts --bind-interfaces --pid-file=/v
5093 1086 Sl /usr/lib/gvfs/gvfsd-http --spawner :1.4 /org/gtk/gvfs/exec_spaw/8
8886 2 S [kworker/1:1]
9259 2 S [kworker/2:2]
9467 2 S [kworker/0:3]
9470 2 S [kworker/u16:46]
10940 2 S [irq/50-mei_me]
10997 709 S /sbin/dhclient -d -q -sf /usr/lib/NetworkManager/nm-dhcp-helper -pf /var/run/dhclient-wlp3s0.
11228 1086 Sl /usr/lib/libreoffice/program/oosplash --writer
11368 2 S [kworker/0:0]
11369 2 S [kworker/u16:0]
11376 2 S [kworker/1:2]
11377 2 S [kworker/3:1]
11380 2 S [kworker/2:0]
11432 11228 Sl /usr/lib/libreoffice/program/soffice.bin
11478 2 S [kworker/1:0]
11480 2 S [kworker/2:1]
11481 2 S [kworker/3:2]
11530 2 S [kworker/u16:1]
11536 2 S [kworker/0:1]
11547 1086 Sl /usr/lib/gnome-terminal/gnome-terminal-server
11554 11547 Ss bash
11573 11554 S+ ./dist/Debug/GNU-Linux/os_processus
11574 11573 Z+ [os_processus] <defunct>
11575 11547 Ss bash
11588 11575 R+ ps -e -o pid,ppid,stat,cmd
jeremie@Extensa-2511:~/Developpement/NetBeansProjects/OS_Exercice_1_1$
```

On constate alors que le processus 11574, lancé par le processus 11573 est en mode Zombie (Z+). En effet, le processus fils envoie un signal indiquant au père qu'il est arrivé au bout de son fil d'exécution, il cherche alors à mourir, mais le père est en mode sommeil et ne peut pas récupérer le signal envoyé par le fils. Le processus fils passe alors en mode zombie.

En interrompant le processus, on constate la disparition du processus père et du processus zombie :

```
jeremie@Extensa-2511: ~/Developpement/NetBeansProjects/OS_Exercice_1_1
1764 1086 Sl /usr/lib/x86_64-linux-gnu/unity-lens-files/unity-files-daemon
1818 1310 Sl /usr/lib/x86_64-linux-gnu/deja-dup/deja-dup-monitor
1834 1086 Sl /usr/lib/gvfs/gvfsd-network --spawner :1.4 /org/gtk/gvfs/exec_spaw/2
1863 1086 Sl /usr/lib/gvfs/gvfsd-dnssd --spawner :1.4 /org/gtk/gvfs/exec_spaw/6
1909 1086 S /usr/lib/x86_64-linux-gnu/gconf/gconfd-2
4718 709 S /usr/sbin/dnsmasq --no-resolv --keep-in-foreground --no-hosts --bind-interfaces --pid-file=/v
5093 1086 Sl /usr/lib/gvfs/gvfsd-http --spawner :1.4 /org/gtk/gvfs/exec_spaw/8
8886 2 S [kworker/1:1]
9470 2 S [kworker/u16:46]
10940 2 S [irq/50-mei_me]
10997 709 S /sbin/dhclient -d -q -sf /usr/lib/NetworkManager/nm-dhcp-helper -pf /var/run/dhclient-wlp3s0.
11228 1086 Sl /usr/lib/libreoffice/program/oosplash --writer
11368 2 S [kworker/0:0]
11432 11228 Sl /usr/lib/libreoffice/program/soffice.bin
11478 2 S [kworker/1:0]
11480 2 S [kworker/2:1]
11530 2 S [kworker/u16:1]
11547 1086 Sl /usr/lib/gnome-terminal/gnome-terminal-server
11554 11547 Ss bash
11627 2 S [kworker/u16:2]
11638 2 S [kworker/2:0]
11640 2 S [kworker/3:1]
11684 2 S [kworker/0:1]
11709 2 S [kworker/2:2]
11713 2 S [kworker/3:2]
11719 2 S [kworker/u16:0]
11740 2 S [kworker/0:2]
11750 2 S [kworker/1:2]
11753 11554 R+ ps -e -o pid,ppid,stat,cmd
jeremie@Extensa-2511:~/Developpement/NetBeansProjects/OS_Exercice_1_1$
```

## Exercice 1.2 :

Le but de cet exercice est de nous faire créer deux processus concurrents qui vont partager des descripteurs de fichier. De cette façon, on pourra rediriger les flux d'entrée sortie (dans notre cas, de sortie) vers un fichier.

### Code source :

#### Header.h

---

```
#ifndef HEADER_H
#define HEADER_H

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

#endif // HEADER_H
```

---

#### Main.c

---

```
#include "headers/header.h"

int main(int argc, char** argv) {
    int pid_filsDate = 0;
    int pid_filsAttente = 0;

    FILE* canal;
    canal = fopen("message.log", "wt");
    if (canal == NULL) {
        printf("%s\n", "Impossible d'ouvrir le fichier message.log");
        exit(0);
    }

    pid_filsDate = fork();
    if (pid_filsDate == 0) {
        for (int i = 0; i < 10; i++) {
            sleep(3);
            pid_filsDate = fork();
```

```
    dup2(canal->_fileno, 1);
    //Redirection de la sortie standard vers message.log.

    if (pid_filsDate == 0) {
        execl("/bin/date", "date", "-u", NULL);
        //execl ecrase le processus qui l'a lancé.
        //Les petits fils ne créeront pas d'autres processus.
    }
}

close(canal->_fileno);
exit(1);
}

pid_filsAttente = fork();
if (pid_filsAttente == 0) {
    for (int i = 0; i < 30; i++) {
        sleep(2);
        pid_filsAttente = fork();

        dup2(canal->_fileno, 1);
        //Redirection de la sortie standard vers message.log.

        if (pid_filsAttente == 0) {
            printf("Attendre\n");
            break; //Seul le fils parcourt la boucle, pas les petits fils.
        }
    }

    close(canal->_fileno);
    exit(2);
}

while (wait(NULL) > -1) {

}

dup2(canal->_fileno, 1);
printf("%s\n", "C'est terminé !");
close(canal->_fileno);
fclose(canal);
}
```

---

## Trace d'exécution :

[illegible]

On constate bien que l'écriture des sortie se fait dans le fichier message.log. De plus, on peut remarquer qu'à chaque appel de la fonction `sleep()`, l'ordonnanceur passe la main à un autre processus, d'où l'enchevêtrement de l'affichage. Le timecode nous montre toutefois que les timings sont bien respectées.

Une fois tous les fils terminé, le processus père quitte en affichant « C'est terminé ! ».

## Chapitre 2 - Communication inter-processus (IPC)

### Exercice 2.1 :

Le but de ce programme est de permettre d'intercepter les signaux émis aux processus pour associer des actions a ces signaux. Dans notre cas, on interceptera les signaux SIGTERM (15 - envoyé par défaut grâce à la commande « kill ») pour incrémenter une variable.

#### Code source :

Header.h :

---

```
#ifndef HEADER_H
#define HEADER_H

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/signal.h>

void handler(int);

#endif // HEADER_H
```

---

Main.c :

---

```
#include "headers/header.h"

int N = 0; //Attention, variable globale !

int main(int argc, char** argv) {
    signal(SIGTERM, handler); //Assignment d'un handler
    while (1) {
        printf("Toc toc %d\n", N);
        sleep(2);
    }
    return (EXIT_SUCCESS);
}

void handler(int sig) {
    if (sig == SIGTERM) {
        N += 1;
        signal(SIGTERM, handler); //Sans cela le programme s'arrête au prochain
                                   //SIGTERM
    } else if (sig == SIGQUIT) { exit(0); }
}
```

---



## Trace d'exécution :

Au lancement du programme, la variable globale N est initialisée à 0. L'affichage de la chaîne de caractère « toc toc » se fait toutes les deux secondes. Chaque fois que le programme reçoit le signal SIGTERM, la variable N est incrémentée.

Sur les deux captures ci dessous, on constate que la variable N est incrémentée autant de fois que l'on appel la commande « kill 3708 ».

```
jeremie@Extensa-2511: ~  
jeremie@Extensa-2511:~$ Developpement/NetBeansProjects/OS_Exercice_2_1/dist/Debug/GNU-Linux/os_exercice_2_1  
Toc toc 0  
Toc toc 0  
Toc toc 0  
Toc toc 0  
Toc toc 0  
Toc toc 0  
Toc toc 0  
Toc toc 0  
Toc toc 0  
Toc toc 0  
Toc toc 0  
Toc toc 0  
Toc toc 0  
Toc toc 0  
Toc toc 1  
Toc toc 2  
Toc toc 3  
Toc toc 4  
Toc toc 4  
Toc toc 4  
Toc toc 4  
Toc toc 5  
Toc toc 6  
Toc toc 7  
Toc toc 8  
Toc toc 9  
Toc toc 9  
Toc toc 9  
Toc toc 10  
Toc toc 10  
Toc toc 10  
^C
```

```
jeremie@Extensa-2511: ~  
root      2874      2  0 15:04 ?        00:00:00 [kworker/u16:1]  
root      2991      2  0 15:05 ?        00:00:00 [kworker/2:0]  
jeremie   3095    1102  2 15:08 ?        00:00:27 /usr/lib/firefox/firefox  
root      3448      2  0 15:13 ?        00:00:00 [kworker/0:2]  
root      3459      2  0 15:13 ?        00:00:00 [kworker/1:0]  
jeremie   3470    1102  0 15:14 ?        00:00:00 /usr/lib/libreoffice/program/oosplash --writer file:///home/j  
jeremie   3489    3470  3 15:14 ?        00:00:17 /usr/lib/libreoffice/program/soffice.bin --writer file:///home  
root      3516      2  0 15:14 ?        00:00:00 [kworker/3:1]  
root      3544      2  0 15:16 ?        00:00:00 [kworker/2:2]  
root      3568      2  0 15:19 ?        00:00:00 [kworker/1:1]  
root      3571      2  0 15:19 ?        00:00:00 [kworker/3:2]  
root      3572      2  0 15:19 ?        00:00:00 [kworker/u16:0]  
root      3586      2  0 15:21 ?        00:00:00 [kworker/2:1]  
root      3592      2  0 15:21 ?        00:00:00 [kworker/0:1]  
jeremie   3685    1102  1 15:23 ?        00:00:00 /usr/lib/gnome-terminal/gnome-terminal-server  
jeremie   3692    3685  0 15:23 pts/0    00:00:00 bash  
jeremie   3708    3692  0 15:23 pts/0    00:00:00 Developpement/NetBeansProjects/OS_Exercice_2_1/dist/Debug/GNU-  
jeremie   3709    3685  1 15:23 pts/17   00:00:00 bash  
jeremie   3720    3709  0 15:23 pts/17   00:00:00 ps -ef  
jeremie@Extensa-2511:~$ kill 3708  
jeremie@Extensa-2511:~$ kill 3708  
jeremie@Extensa-2511:~$ kill 3708  
jeremie@Extensa-2511:~$ kill 3708  
jeremie@Extensa-2511:~$ kill 3708  
jeremie@Extensa-2511:~$ kill 3708  
jeremie@Extensa-2511:~$ kill 3708  
jeremie@Extensa-2511:~$ kill 3708  
jeremie@Extensa-2511:~$ kill 3708  
jeremie@Extensa-2511:~$ kill 3708  
jeremie@Extensa-2511:~$
```

code