Rioufol Simon
Sonmezturk Emre
Luquet Quentin

Compte Rendu TP2-OS302

EXERCICE 1 : Synchronisation père/fils par des signaux

Code :

```c
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
#include <errno.h>
#include <signal.h>
void rien(){};
int main(int argc, char *argv[])
{
    int count = 0;
    if(argc > 1)
    {
        count = atoi(argv[1]); //Atoi renovie 0 pour un chaine sans nombre
        if(count == 0)
            fprintf(stdout, "Attention le paramètre (%s) n'est peut être pas un
nombre.\n"
                    "Le programme va commencer en partant de 0.\n", argv[1]);
        else if(count >= 100)
        {
            fprintf(stderr,"Le nombre donné est trop grand.\n"
                           "Il doit être en dessous de 100.\n");
            return 1;
        }
    }
    pid_t pid = fork(); // création du fils
    if (pid == -1)
        perror("Fork error"), exit(1);
    if(pid) //Pere
    {
        if(count%2) // c'est un nombre impaire le père doit commencer
        {
            fprintf(stdout,"(pere) %d\n",count);
            kill(pid,SIGUSR1);
            count+=2;
```

```
        }
        else
            count++; // C'est un nombre paire donc on rajoute 1 pour avoir le suivant
    }
    else //Fils
    {
        pid = getppid();//On stock le pid du père pour lui envoyer dans la boucle for
qui suit
        if(count%2 == 0) // c'est un nombre paire le fils doit commencer
        {
            fprintf(stdout,"(fils) %d\n",count);
            kill(getppid(),SIGUSR1);
            count+=2;
        }
        else
            count++; // C'est un nombre impaire donc on rajoute 1 pour avoir le
suivant
    }
    for(;count<=100; count+=2)
    {
        signal(SIGUSR1,rien);
        pause();
        fprintf(stdout,"(fils) %d\n",count);
        kill(pid,SIGUSR1);
    }
    return 0;
}
```

Sortie Terminal :

```
ketain@MSI:/mnt/c/Users/quluq/Desktop/OS312/TP2$ ./main
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
```

```
ketain@MSI:/mnt/c/Users/quluq/Desktop/OS312/TP2$ ./main 95
95
96
97
98
99
100
ketain@MSI:/mnt/c/Users/quluq/Desktop/OS312/TP2$ ./main 101
Le nombre donner est trop grands.
Il doit etre en dessous de 100.
ketain@MSI:/mnt/c/Users/quluq/Desktop/OS312/TP2$ ./main qsmdlfjqkl
Attention le parametre (qsmdlfjqkl) n'est peut etre pas un nombre.
Le programme va commencer en partant de 0.
0
1
2
3
4
5
```

EXERCICE 2 : Horloge

Code:

```c
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/types.h>
void signalHandler(int sig){}//ne rien faire
int main()
{
    pid_t minute = fork(); //Le premier fils s'occupe des minutes
    if (minute == -1) {
        fprintf(stderr, "Erreur : fork.\n");
        return EXIT_FAILURE;
    }
    if(minute) //Pere : s'occupe des heures
    {
        fprintf(stdout,"Heure : 00 \n");
        int h = 0;
        while(1)
        {
            signal(SIGUSR1, signalHandler);
            pause();
            fprintf(stdout, "Heure : %d\n", ++h);
            if(h == 60)
                h = 0;
        }
    }
    else // on est dans le premier fils, celui des minutes
    {
        pid_t seconde = fork(); // Le fils du fils minutes s'occupe des secondes
        if (seconde == -1) {
            fprintf(stderr, "Erreur : fork.\n");
            exit(1);
        }
        if(seconde) //On est dans le premier fils, celui des minutes
        {
            sleep(1);
            fprintf(stdout,"Minute : 00\n");
            int min = 0;
            while(1)
            {
```

```
                signal(SIGUSR2, signalHandler);
                pause();
                fprintf(stdout, "min: %d\n", ++min);
                if(min == 60)
                {
                    min = 0;
                    kill(getppid(),SIGUSR1);
                }
            }
        }
        else //On est dans le fils de minutes, celui qui s'occupe des secondes
        {
            sleep(2);
            fprintf(stdout,"Secondes : 00\n");
            int sec = 0;
            while(1)
            {
                alarm(1);
                signal(SIGALRM, signalHandler);
                pause();
                fprintf(stdout,"sec: %d\n",sec++);
                if(sec == 60)
                {
                    sec = 0;
                    kill(getppid(), SIGUSR2);
                }
            }
        }
    }
    return 0;
}
```

Sortie Terminal :

```
ketain@MSI:/mnt/c/Users/quluq/Desktop/OS312/TP2$ ./main
Heure : 00
Minute : 00
Secondes : 00
sec: 0
sec: 1
sec: 2
```

```
sec: 54
sec: 55
sec: 56
sec: 57
sec: 58
sec: 59
min: 1
sec: 0
sec: 1
sec: 2
sec: 3
sec: 4
sec: 5
sec: 6
sec: 7
```

EXERCICE 3 : Roulette russe

Code :

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

#define TAILLE_MAX 20
int valeur;
int finJeu = 0;
void rien(){} //rien faire

void signalHandler(int sig)
{
    switch(sig)
    {
        case SIGUSR1: //Regarder dans le fichier
            fprintf(stdout,"Je suis %d et je vais essayer de ne pas mourir. \n",
getpid());
            FILE *fp = fopen("Barillet", "r");
            if(fp == NULL)
                exit(1);
            char chaine[TAILLE_MAX];
            fgets(chaine,TAILLE_MAX, fp);
            fclose(fp);
            valeur = atoi(chaine);
            fprintf(stdout,"Fin de la lecture le num est %d.\n",valeur);
            break;
        case SIGUSR2: // Si un fils est mort go mourir en gros
            fprintf(stdout, "(%d) Je dois m'auto detruire malgré ma victoire. ;(  \n",
getpid());
            exit(1);
            break;
        case SIGCHLD:
            finJeu = 1;
            break;
        default:
            break;
    }
}
/*
```

```c
 * Fonction qui crée un nombre défini de processus fils en leurs associant chacun un
numero
 * On stock tout les PID des fils crée dans un tableau
 */
void forkNB(pid_t pidChild[], int *num, int nb_fils)
{
  int numero = 0;
  do{
    numero++;
    pidChild[numero-1] = fork();

    if (pidChild[numero-1] == -1) {
        fprintf(stderr, "Erreur : fork.\n");
        exit(1);
    }
  }while(pidChild[numero-1] != 0 && numero < nb_fils);

  if(pidChild[nb_fils-1] != 0) // Attribution du numero des processus, 0 est le père.
    *num = 0;
  else
    *num = numero;
}

int main()
{
    int nb_fils = 200;
    pid_t *pidFils = malloc(sizeof(pid_t)*nb_fils);
    ////////////////////////Ne pas oublier de free ce tableau ///////////////////
    if(pidFils == NULL)
        fprintf(stderr, "Erreur malloc.\n"), exit(1);
    int numeroProcessus = 0;  // Num propre à chaque processus 0 pour le père, 1 pour
le 1er fils
    forkNB(pidFils, &numeroProcessus, nb_fils);

    if(!numeroProcessus)
    {
        sleep(1); // temps d'attente de tous les processus soit en pause()
        int processMort = 0, i = 0;
        while(i < nb_fils)
        {
            fprintf(stdout,"(Pere) Envoie du signal SIGUSR1 à %d.\n", pidFils[i]);
            kill(pidFils[i], SIGUSR1);
            signal(SIGCHLD, signalHandler);
```

```
            signal(SIGUSR2, rien);
            pause();
            if(finJeu)
            {
                fprintf(stdout, "(Pere) Le process %d est mort.\n", i+1);
                processMort = i;
                i = nb_fils;
            }
            i++;
        }

        for(i = 0; i<nb_fils;i++)
        {
            if(!finJeu)//mettre fin a tout les processus mais situation bizarre
            {
                kill(pidFils[i], SIGUSR2);
                fprintf(stdout, "(Pere) Extermination num = %d.\n", i+1);
            }
            else if(i != processMort)
            {
                kill(pidFils[i], SIGUSR2);
                fprintf(stdout, "(Pere) Extermination num = %d.\n", i+1);
            }
        }
        free(pidFils);
        exit(1);
    }
    else
    {
        signal(SIGUSR1,signalHandler);
        signal(SIGUSR2,signalHandler);
        pause();
        if(valeur == numeroProcessus)
        {
            fprintf(stdout, "Oh non je meurs (numLu = %d, num = %d).\n", valeur,
numeroProcessus);
            exit(1);
        }
        else
        {
            fprintf(stdout ,"Oof pas mort (numLu = %d, num = %d).\n", valeur,
numeroProcessus);
            kill(getppid(),SIGUSR2); // au suivant
```

```
            pause();
        }
    }
}
```

Sortie Terminal :

```
ketain@MSI:/mnt/c/Users/quluq/Desktop/OS312/TP2$ cat Barillet
4
ketain@MSI:/mnt/c/Users/quluq/Desktop/OS312/TP2$ ./main
(Pere) Envoie du signal SIGUSR1 à 271.
Je suis 271 et je vais essayer de ne pas mourir.
Fin de la lecture le num est 4.
Oof pas mort (numLu = 4, num = 1).
(Pere) Envoie du signal SIGUSR1 à 272.
Je suis 272 et je vais essayer de ne pas mourir.
Fin de la lecture le num est 4.
Oof pas mort (numLu = 4, num = 2).
(Pere) Envoie du signal SIGUSR1 à 273.
Je suis 273 et je vais essayer de ne pas mourir.
Fin de la lecture le num est 4.
Oof pas mort (numLu = 4, num = 3).
(Pere) Envoie du signal SIGUSR1 à 274.
Je suis 274 et je vais essayer de ne pas mourir.
Fin de la lecture le num est 4.
Oh non je meurt (numLu = 4, num = 4).
(Pere) Le process 4 est mort.
(Pere) Extermination num = 1.
(271) Je dois m'auto detruire malgré ma victoire. ;(
(Pere) Extermination num = 2.
(272) Je dois m'auto detruire malgré ma victoire. ;(
(Pere) Extermination num = 3.
(273) Je dois m'auto detruire malgré ma victoire. ;(
(Pere) Extermination num = 5.
(275) Je dois m'auto detruire malgré ma victoire. ;(
(Pere) Extermination num = 6.
(276) Je dois m'auto detruire malgré ma victoire. ;(
```