TP n° 2 – éléments de réponse

"sin_cos"

```c
/**
 * @file sin_cos.h
 */

#ifndef __SIN_COS_H
#define __SIN_COS_H

#ifndef M_PI
#define M_PI 3.14159
#endif

#define EVER ;;

void increment_angle();
void sinus(int i);
void cosinus(int i);
int manage_child_cos();
int manage_child_sin();

#endif
```

```c
/**
 * @file sin_cos.c
 */

#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <math.h>
#include <string.h>
#include "sin_cos.h"

int angle = 0;

void increment_angle()
{
```

```
    angle += 10;
    if (angle == 360) {
        angle = 0;
    }
}

void sinus(int i)
{
    increment_angle();
    printf("sinus(%d)=%.2f\n", angle, sin(angle * 2 * M_PI / 360));
    alarm(1);
}

void cosinus(int i)
{
    increment_angle();
    printf("cosinus(%d)=%.2f\n", angle, cos(angle * 2 * M_PI / 360));
    alarm(1);
}

int manage_child_cos()
{
    struct sigaction action;

    memset(&action, 0, sizeof(action));
    action.sa_handler = &cosinus;
    if (sigaction(SIGALRM, &action, NULL) == -1) {
        return -1;
    }
    alarm(1);
    return 0;
}

int manage_child_sin()
{
    struct sigaction action;

    memset(&action, 0, sizeof(action));
    action.sa_handler = &sinus;
    if (sigaction(SIGALRM, &action, NULL) == -1) {
        return -1;
    }
```

```
    alarm(1);
    return 0;
}
```

```
/**
 * @file main.c
 */

#include <stdio.h>
#include <unistd.h>
#include "sin_cos.h"

extern int angle;

int main(void)
{
    if (fork() == 0) {
        manage_child_sin();
    } else if (fork() == 0) {
        manage_child_cos();
    }
    /* father and its two childs infinitly work */
    for (EVER) ;
}
```

"sin_cos_g"

```
/**
 * @file sin_cos_g.h
 */

#ifndef SIN_COS_G_H
#define SIN_COS_G_H

#ifndef M_PI
#define M_PI 3.14159
#endif

#define ROW_SIZE 11
```

```
void plot();
void sinus();
void cosinus(int signum);
void install_handlerPlot();
void install_handler_sin();
void install_handler_cos();

#endif
```

```
/**
 * @file sin_cos_2.c
 */

#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>
#include <math.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <fcntl.h>
#include <string.h>
#include "sin_cos_2.h"

int angle1;
int angle2;
pid_t pid1;
pid_t pid2;
FILE * pidFilehandle1;
FILE * pidFilehandle2;

void plot()
{
    int status;
    FILE *fq;

    printf("> Prepare plotting using Gnuplot\n");

    /* Spawn a gnuplot process to plot data file */
    if (fork() == 0) {
```

```
        fq = fopen("commands1.gp", "w");
        fprintf(fq, "plot \"sin.txt\" using 1:2 smooth csplines\n");
        fclose(fq);

        if (execlp("gnuplot", "gnuplot", "-persist", "./commands1.gp", (void *)0)
        ↪  < 0) {
            fprintf(stderr, "Error!\n");
            exit(EXIT_FAILURE);
        }
    } else {
        /* Wait for child to exit, and store child's exit status */
        wait(&status);
        printf("Child exit code: %d\n", WEXITSTATUS(status));

        /* Spawn a gnuplot process to plot data file */
        if (fork() == 0) {
            fq = fopen("commands2.gp", "w");
            fprintf(fq, "plot \"cos.txt\" using 1:2 smooth csplines\n");
            fclose(fq);

            if (execlp("gnuplot", "gnuplot", "-persist", "./commands2.gp", (void
            ↪  *)0) < 0) {
                fprintf(stderr, "Error!\n");
                exit(EXIT_FAILURE);
            }
        } else {
            /* wait for child to exit, and store child's exit status */
            wait(&status);
            printf("Child exit code: %d\n", WEXITSTATUS(status));
        }
    }
    exit(EXIT_SUCCESS);
}

void sinus()
{
    char row[ROW_SIZE];

    if (angle1 == 360) {
        angle1 = 0;
        alarm(0);
        fclose(pidFilehandle1);
```

```
        exit(EXIT_SUCCESS);
    }
    /* format gnuplot row */
    sprintf(row, "%d\t%.2f\n", angle1, sin(angle1 * 2 * M_PI / 360));
    fprintf(pidFilehandle1, "%s", row);
    fflush(pidFilehandle1);
    angle1 += 10;
    alarm(1);
}

void cosinus(int signum)
{
    char row[ROW_SIZE];

    if (angle2 == 360) {
        angle2 = 0;
        alarm(0);
        fflush(pidFilehandle2);
        fclose(pidFilehandle2);
        exit(EXIT_SUCCESS);
    }
    /* format gnuplot row */
    sprintf(row, "%d\t%.2f\n", angle2, cos(angle2 * 2 * M_PI / 360));
    fprintf(pidFilehandle2, "%s", row);
    angle2 += 10;
    alarm(1);
}

void install_handlerPlot()
{
    struct sigaction action;

    memset(&action, 0, sizeof(action));
    action.sa_handler = &plot;
    sigaction(SIGCHLD, &action, NULL);
}

void install_handler_sin()
{
    struct sigaction actionPID;

    memset(&actionPID, 0, sizeof(actionPID));
```

```
    actionPID.sa_handler = &sinus;
    sigaction(SIGALRM, &actionPID, NULL);
}

void install_handler_cos()
{
    struct sigaction actionPID;

    memset(&actionPID, 0, sizeof(actionPID));
    actionPID.sa_handler = &cosinus;
    sigaction(SIGALRM, &actionPID, NULL);
}
```

```
/**
 * @file main.c
 */

#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>
#include <math.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <fcntl.h>
#include <string.h>
#include "sin_cos_2.h"

extern int angle1;
extern int angle2;
extern pid_t pid1;
extern pid_t pid2;
extern FILE * pidFilehandle1;
extern FILE * pidFilehandle2;

int main(void)
{
    install_handlerPlot();

    if (fork() == 0) {
        pid1 = getpid();
```

```
        pidFilehandle1 = fopen("sin.txt", "w");
        install_handler_sin();
        alarm(1);
    } else {
        if (fork() == 0) {
            pid2 = getpid();
            pidFilehandle2 = fopen("cos.txt", "w");
            install_handler_cos();
            alarm(1);
        }
    }
    while (wait(NULL) != pid1 && wait(NULL) != pid2)
        ;

    exit(EXIT_SUCCESS);
}
```