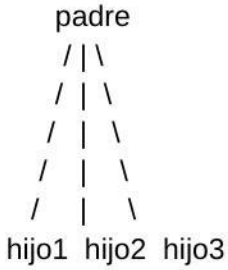
# Solución. Práctica 4. Creación, Comunicación y Sincronización de procesos

1. Implementa  un programa que cree un hijo. El hijo debe escribir “Soy el hijo” diez veces. El padre debe escribir “Soy el padre” diez veces. El padre debe esperar a que termine el hijo y mostrar el mensaje “Mi proceso hijo ya ha terminado”.

```c
/* UT01.Programacion Multiproceso */
/* Practica 4. ejercicio1 */


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

int main(int argc, char *argv[])
{
        pid_t pid;
        int status;
        pid = fork();
        int i;

        if (pid == 0)/*hijo*/
        {
                for(i=1;i<=10;i++){
                        printf("Soy el hijo (%d, hijo de %d)\n",getpid(),getppid());
                }
        }
        else/*padre*/
        {
                waitpid(pid, &status, 0); // padre espera a hijo

                for(i=1;i<=10;i++){
                        printf("Soy el padre (%d, hijo de %d)\n",getpid(),getppid());
                }

        }
        return 0;
}
```

```
david@david-OEM ~/Escritorio/Abastos/psp/ut1/sol_pract4 $ gcc 4_1.c -o 1
david@david-OEM ~/Escritorio/Abastos/psp/ut1/sol_pract4 $ ./1
Soy el hijo (4072, hijo de 4071)
Soy el hijo (4072, hijo de 4071)
Soy el hijo (4072, hijo de 4071)
Soy el hijo (4072, hijo de 4071)
Soy el hijo (4072, hijo de 4071)
Soy el hijo (4072, hijo de 4071)
Soy el hijo (4072, hijo de 4071)
Soy el hijo (4072, hijo de 4071)
Soy el hijo (4072, hijo de 4071)
Soy el hijo (4072, hijo de 4071)
Soy el padre (4071, hijo de 3957)
Soy el padre (4071, hijo de 3957)
Soy el padre (4071, hijo de 3957)
Soy el padre (4071, hijo de 3957)
Soy el padre (4071, hijo de 3957)
Soy el padre (4071, hijo de 3957)
Soy el padre (4071, hijo de 3957)
Soy el padre (4071, hijo de 3957)
Soy el padre (4071, hijo de 3957)
Soy el padre (4071, hijo de 3957)
```

2. Realice un programa en C que genere la siguiente configuración de procesos:

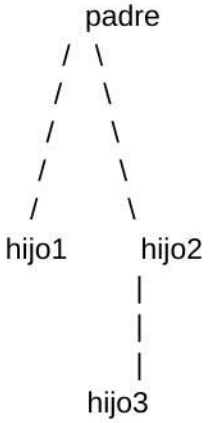



Además, cada hijo deberá mostrar el mensaje "Yo soy el hijo x, mi padre es PID=Y, yo soy PID=z".

```c
int main(int argc, char *argv[])
{
        pid_t pid1,pid2,pid3;
        int status;
        pid1 = fork();
        int i;

        if (pid1 == 0)/*hijo*/
        {

                printf("Soy el hijo 1 (%d, hijo de %d)\n",getpid(),getppid());

        }

        else /*de hijo1*/
        {
                pid2 = fork(); //creo hijo2

                if (pid2 == 0)/*hijo2*/
                {

                printf("Soy el hijo2 (%d, hijo de %d)\n",getpid(),getppid());

                }

                else /*de hijo2*/
                {
                        pid3 = fork(); //creo hijo3

                        if (pid3 == 0)/*hijo3*/
                        {

                        printf("Soy el hijo3 (%d, hijo de %d)\n",getpid(),getppid());

                        }
                        else /*de hijo 3*/
                        {
                        //padre de todos
                        waitpid(pid1, &status, 0); // padre espera a hijo1
                        waitpid(pid2, &status, 0); // padre espera a hijo2
                        waitpid(pid3, &status, 0); // padre espera a hijo2
                        printf("Soy el padre (%d, hijo de %d)\n",getpid(),getppid());

                        }//else de tercer hijo
                } //else segundo hijo

        } //else primer hijo
        return 0;
}
```

```
david@david-OEM ~/Escritorio/Abastos/psp/ut1/sol_pract4 $ ./2
Soy el hijo 1 (4440, hijo de 4439)
Soy el hijo3 (4442, hijo de 4439)
Soy el hijo2 (4441, hijo de 4439)
Soy el padre (4439, hijo de 3957)
```

3. Realiza lo mismo que en el ejercicio anterior pero con la siguiente configuración de procesos:



```c
if (pid1 == 0)/*hijo*/
{

        printf("Soy el hijo 1 (%d, hijo de %d)\n",getpid(),getppid());

}

else /*de hijo1*/
{
        pid2 = fork(); //creo hijo2

        if (pid2 == 0)/*hijo2*/
        {

        printf("Soy el hijo2 (%d, hijo de %d)\n",getpid(),getppid());

        //creo hijo 3
        pid3 = fork(); //creo hijo3

                if (pid3 == 0)/*hijo3*/
                {

                printf("Soy el hijo3 (%d, hijo de %d)\n",getpid(),getppid());

                }
        waitpid(pid3, &status, 0); // hijo2 espera a su hijo --> hijo3
        }

        else /*de hijo2*/
        {

                //padre de todos
                waitpid(pid1, &status, 0); // padre espera a hijo1
                waitpid(pid2, &status, 0); // padre espera a hijo2

                printf("Soy el padre (%d, hijo de %d)\n",getpid(),getppid());


        } //else segundo hijo

} //else primer hijo
return 0;
```

```
david@david-OEM ~/Escritorio/Abastos/psp/ut1/sol_pract4 $ ./3
Soy el hijo 1 (4509, hijo de 4508)
Soy el hijo2 (4510, hijo de 4508)
Soy el hijo3 (4511, hijo de 4510)
Soy el padre (4508, hijo de 3957)
```

4. Crea un programa que permita elegir el orden de ejecución de dos procesos p1 y p2, mediante la orden introducida en el terminal.

   Ejemplo

   Introduzco >1
   > Soy proceso 1 y termino
   > Soy proceso 2 y termino

   Introduzco>2
   >Soy proceso 2 y termino
   >Soy proceso 1 y termino

```c
int main(int argc, char * argv[]) {
        int pid1, pid2, estado;
        int status;
        printf("Introduce opcion : 1 / 2 \n");
        int opcion;
        scanf("%d", &opcion);
        /*garantizamos hijo siempre acabe antes padre con waitpid*/
        switch(opcion){
                case 1:
                        pid1 = fork();
                        if (pid1 == 0)/*hijo*/
                                {
                                printf("Soy proceso 1 y termino\n");
                                }

                        else /*padre*/
                                {
                                waitpid(pid1, &status, 0); // padre espera a hijo
                                printf("Soy proceso 2 y termino\n");
                                }


                        break;
                case 2:
                        pid2 = fork();
                        if (pid2 == 0)/*hijo*/
                                {
                                printf("Soy proceso 2 y termino\n");
                                }

                        else /*padre*/
                                {
                                waitpid(pid2, &status, 0); // padre espera a hijo
                                printf("Soy proceso 1 y termino\n");
                                }
                        break;

                default:
                        printf("Opción no válida\n");
                        break;
        }
```

```
david@david-OEM ~/Escritorio/Abastos/psp/ut1/sol_pract4 $ ./4
Introduce opcion : 1 / 2
2
Soy proceso 2 y termino
Soy proceso 1 y termino
david@david-OEM ~/Escritorio/Abastos/psp/ut1/sol_pract4 $ ./4
Introduce opcion : 1 / 2
1
Soy proceso 1 y termino
Soy proceso 2 y termino
david@david-OEM ~/Escritorio/Abastos/psp/ut1/sol_pract4 $ ./4
Introduce opcion : 1 / 2
3
Opción no válida
```