3. Ejercicio

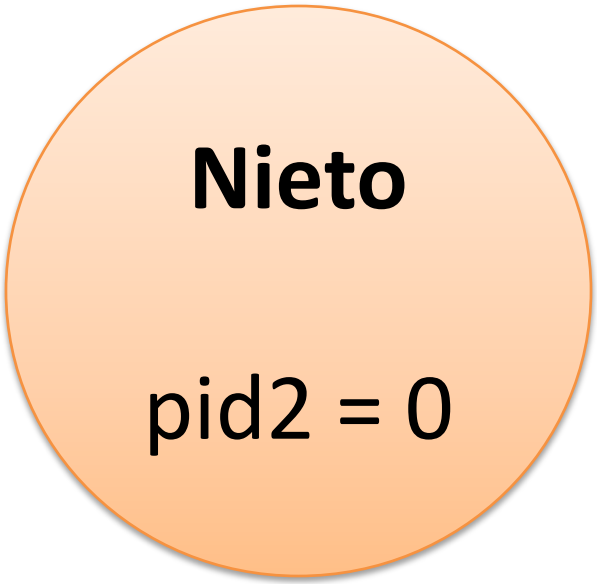
Creamos un nuevo proceso en el proceso hijo; así tendremos el proceso padre (ABUELO), el proceso hijo (HIJO) y el proceso hijo del hijo (NIETO).

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

void main () {

    pid_t pid, Hijo_pid, pid2, Hijo2_pid;
```

```
pid = fork(); //Soy el Abuelo, creo a Hijo
```

```
if (pid == -1
{
   //Ha ocurrido un error
   printf("No se ha podido crear el proceso hijo
   ... ");

   exit( -1);
}
```

```
if (pid == 0 ) //Nos encontramos en Proceso hijo
{
    pid2 = fork(); //Soy el Hijo, creo a Nieto
```



```
switch(pid2)
{
    case -1: / / error
    printf("No se ha podido crear el proceso hijo
            en el HIJO ... ");

    exit(-1);

    break;
```

```
case 0: // proceso hijo
    printf("\t\t Soy el proceso NIETO %d; Mi
    padre es = %d \n ", getpid(), getppid());

    break;
```

```
    default: // proceso padre
        Hijo2_pid = wait(NULL);

        printf("\tSoy el proceso HIJO %d, Mi padre
        es: %d.\n", getpid(), getppid());

        printf("\tMi hijo: %d terminó.\n",
        Hijo2_pid);
    }
}
```

```
else //Nos encontramos en Proceso padre
{
    Hijo_pid = wait(NULL); //espera la
 finalización del proceso hijo

   printf("Soy el proceso ABUELO: %d, Mi HIJO:
 %d terminó.\n", getpid (), pid) ;
 }
```

```
    exit(0);
}
```

La compilación y ejecución:

mj@ubuntu-mj:-$ gcc ejemplo1_2Fork.c -o ejemplo1 2Fork

mj@ubuntu-mj:-$ ./ejemplo1 2Fork

Muestra la siguiente salida:

Soy el proceso NIETO 4486; Mi padre es= 4485

Soy el proceso HIJO 4485, Mi padre es: 4484.
Mi hijo: 4486 terminó.

Soy el proceso ABUELO: 4484, Mi HIJO: 4485 terminó.
mj@ubuntu-mj :-$