

## 6.- Ejercicios a realizar:

1. Observa el siguiente código y escribe la jerarquía de procesos resultante.

```
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
int main (int argc, char *argv[]) {
    int num;
    pid_t pid;
    for (num= 0; num< 3; num++) { pid= fork();
    printf ("Soy el proceso de PID %d y mi padre tiene%d de PID.\n",
    getpid(), getppid());
    if (pid!= 0)
    break;
    srandom(getpid());
    sleep (random() %3);
    }if (pid!= 0)
    printf ("Fin del proceso de PID %d.\n", wait (NULL));
    return 0;
}
```

Ahora compila y ejecuta el código para comprobarlo. Contesta a las siguientes preguntas:

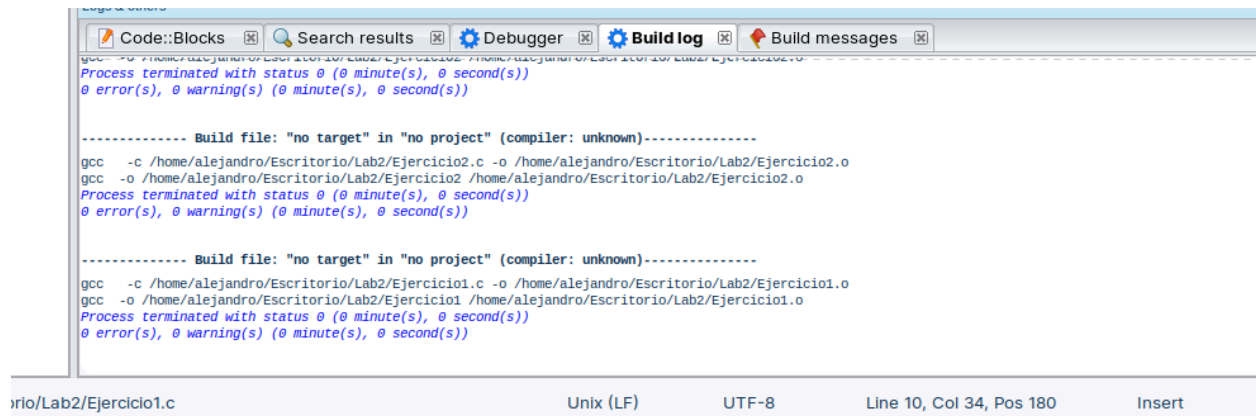
¿Por qué aparecen mensajes repetidos?

Presta atención al orden de terminación de los procesos,

- ¿Qué observas?
- ¿Por qué?

## RESPUESTA:

## CORRIDA Y COMPILACIÓN

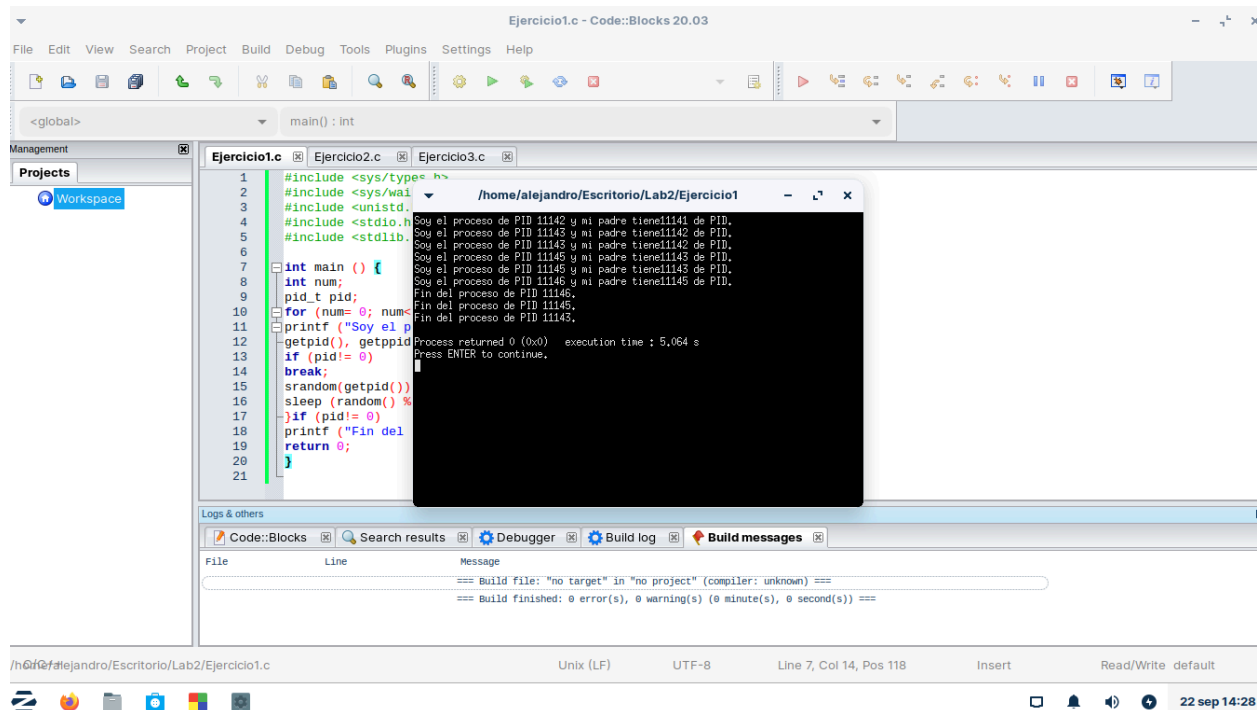


```
Code::Blocks Search results Debugger Build log Build messages
Process terminated with status 0 (0 minute(s), 0 second(s))
0 error(s), 0 warning(s) (0 minute(s), 0 second(s))

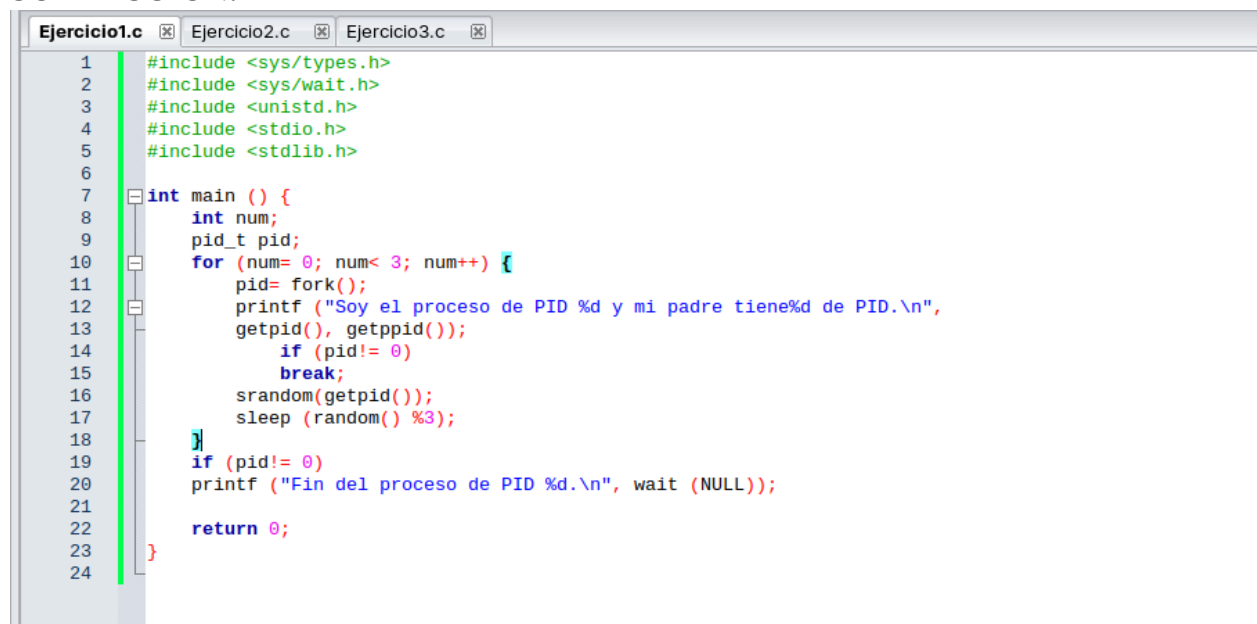
----- Build file: "no target" in "no project" (compiler: unknown)-----
gcc -c /home/alejandro/Escritorio/Lab2/Ejercicio2.c -o /home/alejandro/Escritorio/Lab2/Ejercicio2.o
gcc -o /home/alejandro/Escritorio/Lab2/Ejercicio2 /home/alejandro/Escritorio/Lab2/Ejercicio2.o
Process terminated with status 0 (0 minute(s), 0 second(s))
0 error(s), 0 warning(s) (0 minute(s), 0 second(s))

----- Build file: "no target" in "no project" (compiler: unknown)-----
gcc -c /home/alejandro/Escritorio/Lab2/Ejercicio1.c -o /home/alejandro/Escritorio/Lab2/Ejercicio1.o
gcc -o /home/alejandro/Escritorio/Lab2/Ejercicio1 /home/alejandro/Escritorio/Lab2/Ejercicio1.o
Process terminated with status 0 (0 minute(s), 0 second(s))
0 error(s), 0 warning(s) (0 minute(s), 0 second(s))

rio/Lab2/Ejercicio1.c Unix (LF) UTF-8 Line 10, Col 34, Pos 180 Insert
```



## CORRECCIÓN:



### ¿Por qué aparecen mensajes repetidos?

Porque los mensajes los imprimen tanto los procesos padres como los procesos hijos.

Como podemos ver en la ejecución del programa (Ejecución y compilación distinta a las imágenes anteriores )

Los mensajes repetidos representan la función del fork() la cual es crear una copia del proceso actual y ambos procesos (el padre y el hijo) continúan ejecutando el código a partir de ese punto.

```
/home/alejandro/Escritorio/Lab2/Ejercicio1
Soy el proceso de PID 11326 y mi padre tiene11325 de PID.
Soy el proceso de PID 11327 y mi padre tiene11326 de PID.
Soy el proceso de PID 11327 y mi padre tiene11326 de PID.
Soy el proceso de PID 11328 y mi padre tiene11327 de PID.
Soy el proceso de PID 11328 y mi padre tiene11327 de PID.
Soy el proceso de PID 11329 y mi padre tiene11328 de PID.
Fin del proceso de PID 11329.
Fin del proceso de PID 11328.
Fin del proceso de PID 11327.

Process returned 0 (0x0)   execution time : 4.020 s
Press ENTER to continue.
█
```

El Proceso 11326 es hijo del 11325 a su vez el proceso 11327 es hijo del proceso 11326 que a su vez es padre del proceso 11327 que a su vez es padre del proceso 11328 y este es padre del proceso 11329 haciendo una jerarquía de procesos como la siguiente:

11325 ->11326->11327->11328->11329

**Presta atención al orden de terminación de los procesos,**

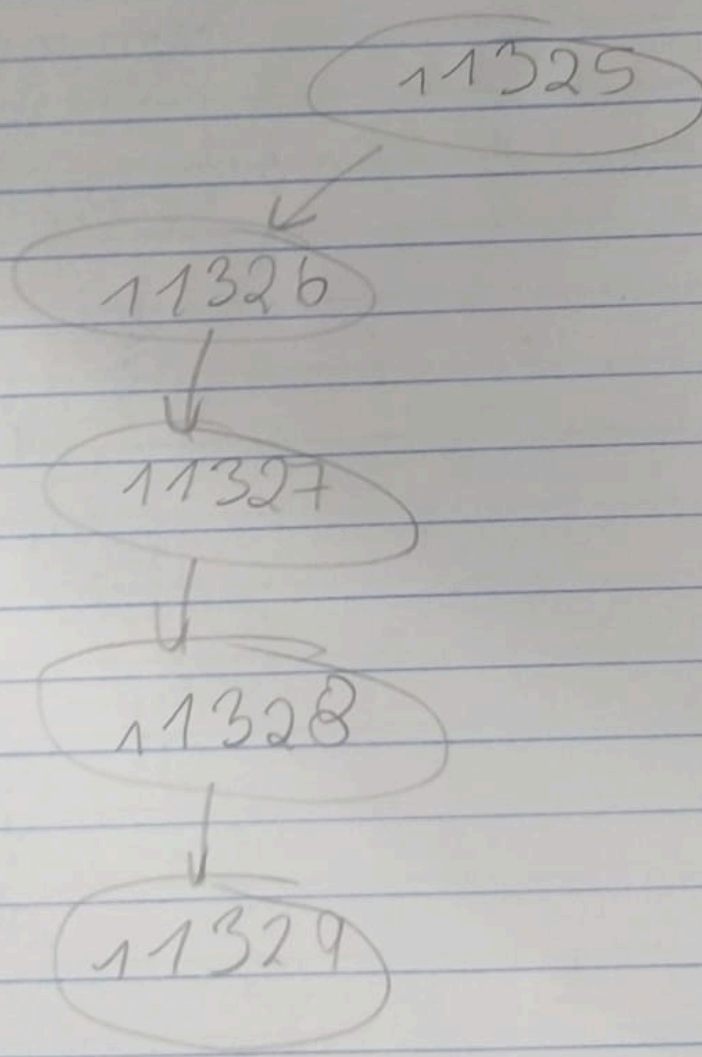
- **¿Qué observas?**

Puedo observar que del bucle for crea 3 procesos hijos a partir del proceso padre

- **¿Por qué?**

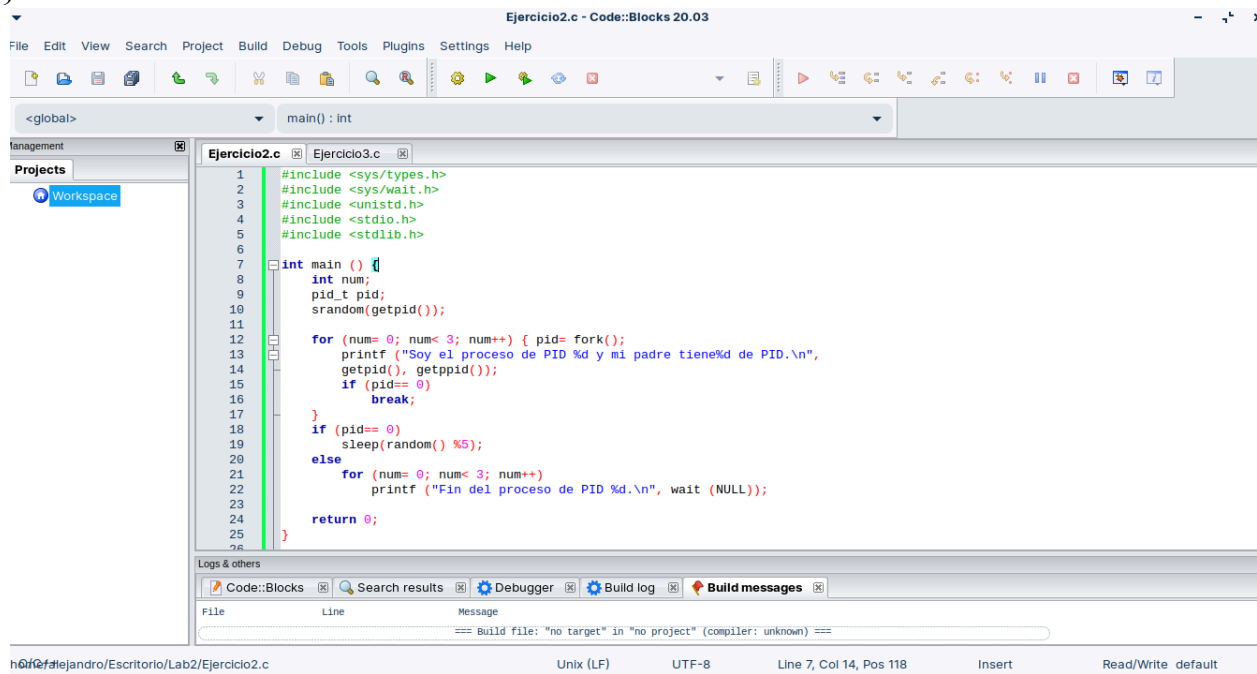
Los procesos se terminan de forma ascendente empezando de los procesos hijos hacia los padres

1



2. Observa el siguiente código y escribe la jerarquía de procesos resultante.

```
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
int main (int argc, char *argv[]) {
    int num;
    pid_t pid;
    srand(getpid());
    for (num= 0; num< 3; num++) { pid= fork();
    printf ("Soy el proceso de PID %d y mi padre tiene%d de PID.\n",
    getpid(), getppid());
    if (pid== 0)
        break;
    }if (pid== 0)
    sleep(random() %5);
    else
    for (num= 0; num< 3; num++)
    printf ("Fin del proceso de PID %d.\n", wait (NULL));
    return 0;
}
```



The screenshot shows a code editor with a C program. The program uses a `for` loop to create three child processes, each printing its own PID and its parent's PID. A terminal window is open, showing the output of the program. The output shows three child processes (PIDs 11601, 11602, 11603) all reporting their parent as PID 11600. This indicates that the child processes are not receiving the updated parent PID from the loop. The terminal also shows the program returning 0 and the execution time being 1.002 seconds.

```
10 // ...
11 // ...
12 for (num= 0; num< 3; num++) { pid= fork();
13     printf ("Soy el proceso de PID %d y mi padre tiene %d de PID \n", pid, pid);
    // ...
}

// ...

le: "no target" in "no project" (compiler: unknown)-----
gcc -c /home/alejandros/Escritorio/Lab2/Ejercicio2.c -o /home/alejandros/Escritorio/Lab2/Ejercicio2.o
gcc -o /home/alejandros/Escritorio/Lab2/Ejercicio2 /home/alejandros/Escritorio/Lab2/Ejercicio2.o
Process terminated with status 0 (0 minute(s), 0 second(s))
0 error(s), 0 warning(s) (0 minute(s), 0 second(s))

le: "no target" in "no project" (compiler: unknown)-----
gcc -c /home/alejandros/Escritorio/Lab2/Ejercicio2.c -o /home/alejandros/Escritorio/Lab2/Ejercicio2.o
gcc -o /home/alejandros/Escritorio/Lab2/Ejercicio2 /home/alejandros/Escritorio/Lab2/Ejercicio2.o
Process terminated with status 0 (0 minute(s), 0 second(s))
0 error(s), 0 warning(s) (0 minute(s), 0 second(s))

/home/alejandros/Escritorio/Lab2/Ejercicio2
/home/alejandros/Escritorio/Lab2/Ejercicio2' -e '/usr/bin/cb_console_runner' '/home/alejandros/Escritorio/Lab2/Ejercicio2' (in '/home/alejandros/Escritorio/Lab2/Ejercicio2')
```

Ahora compila y ejecuta el código para comprobarlo. Presta atención al orden de terminación de los procesos,  
¿Qué observas? ¿Por qué?

### RESPUESTA:

Ahora compila y ejecuta el código para comprobarlo. Presta atención al orden de terminación de los procesos, (Ejecución y compilación distinta a las imágenes anteriores)

```
/home/alejandro/Escritorio/Lab2/Ejercicio2
Soy el proceso de PID 11854 y mi padre tiene11853 de PID.
Soy el proceso de PID 11855 y mi padre tiene11854 de PID.
Soy el proceso de PID 11854 y mi padre tiene11853 de PID.
Soy el proceso de PID 11856 y mi padre tiene11854 de PID.
Soy el proceso de PID 11854 y mi padre tiene11853 de PID.
Soy el proceso de PID 11857 y mi padre tiene11854 de PID.
Fin del proceso de PID 11855.
Fin del proceso de PID 11856.
Fin del proceso de PID 11857.

Process returned 0 (0x0)   execution time : 4.005 s
Press ENTER to continue.
█
```

¿Qué observas?

A diferencia del ejercicio anterior puedo ver que los procesos hijos son terminados en el orden en que fueron creados

¿Por qué?

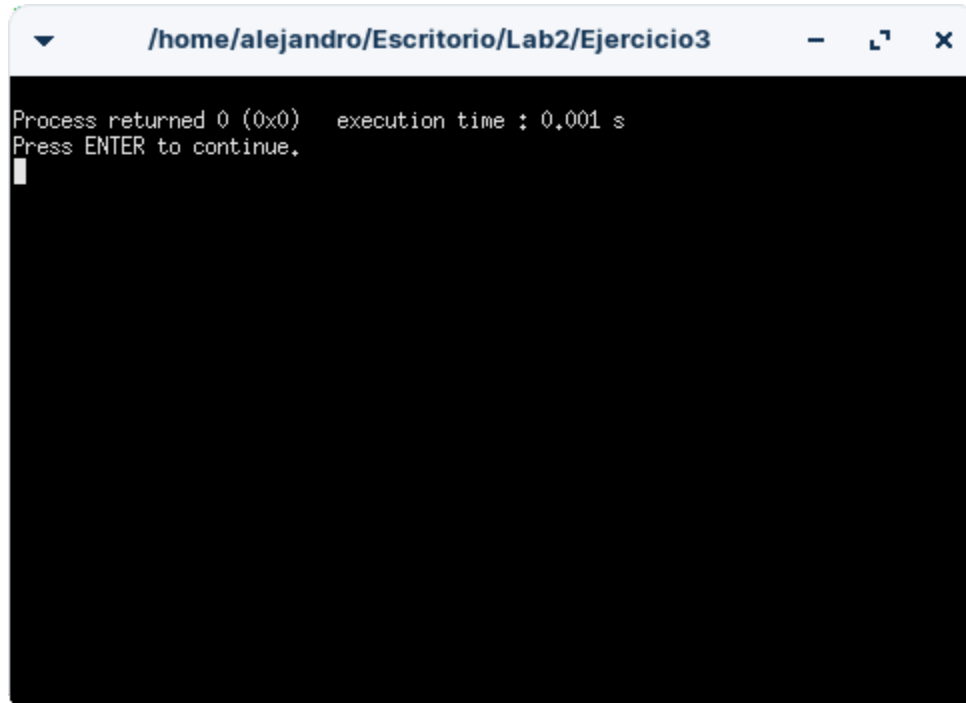
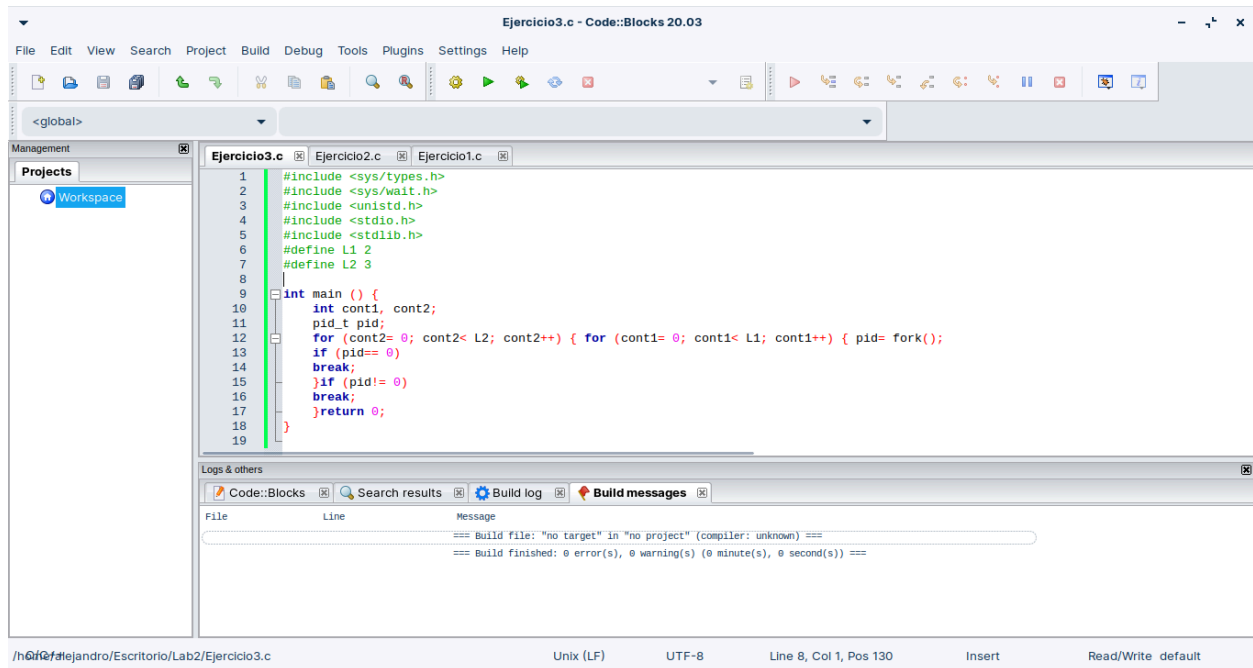
Porque el proceso padre está esperando explícitamente a que todos sus hijos terminen antes de continuar la salida seguirá el orden en el que se crearon los procesos hijos

3. Estudia el siguiente código y escribe la jerarquía de procesos resultante. Después, compila y ejecuta el código para comprobarlo (deberás añadir llamadas al sistema *getpid*, *getppid* y *wait* para conseguirlo).

```
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#define L1 2
#define L2 3
int main (int argc, char *argv[]) {
    int cont1, cont2;
    pid_t pid;
    for (cont2= 0; cont2< L2; cont2++) { for (cont1= 0; cont1< L1; cont1++) { pid= fork();
    if (pid== 0)
        break;
    }if (pid!= 0)
        break;
```

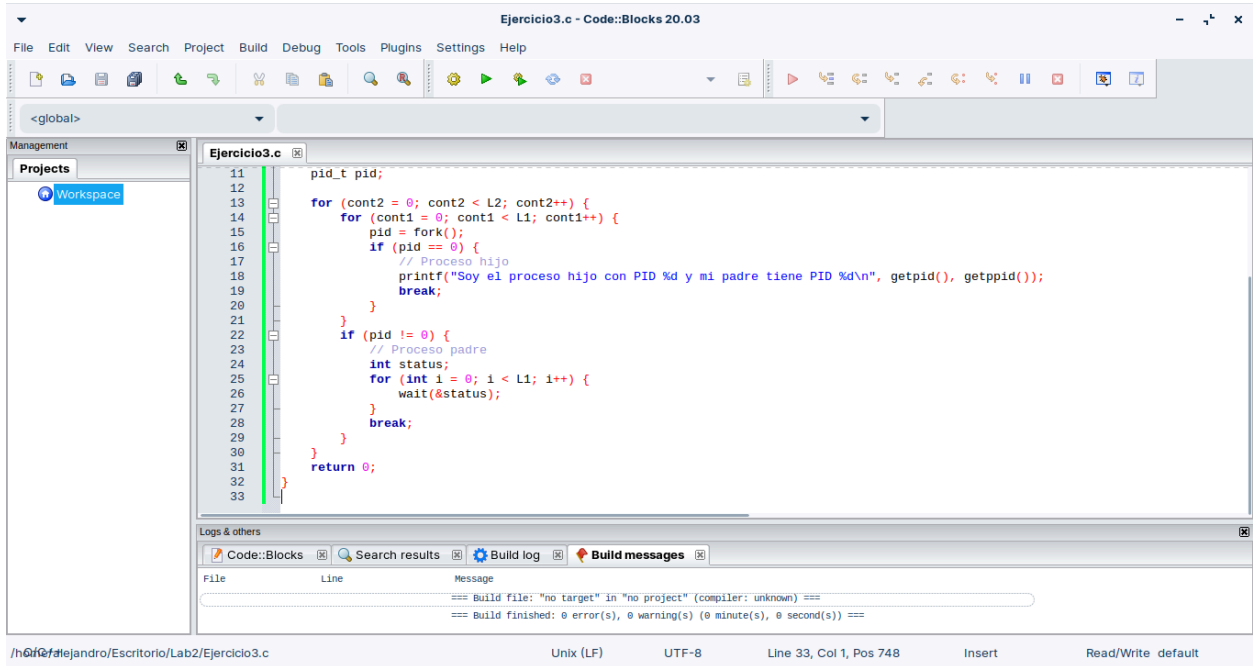
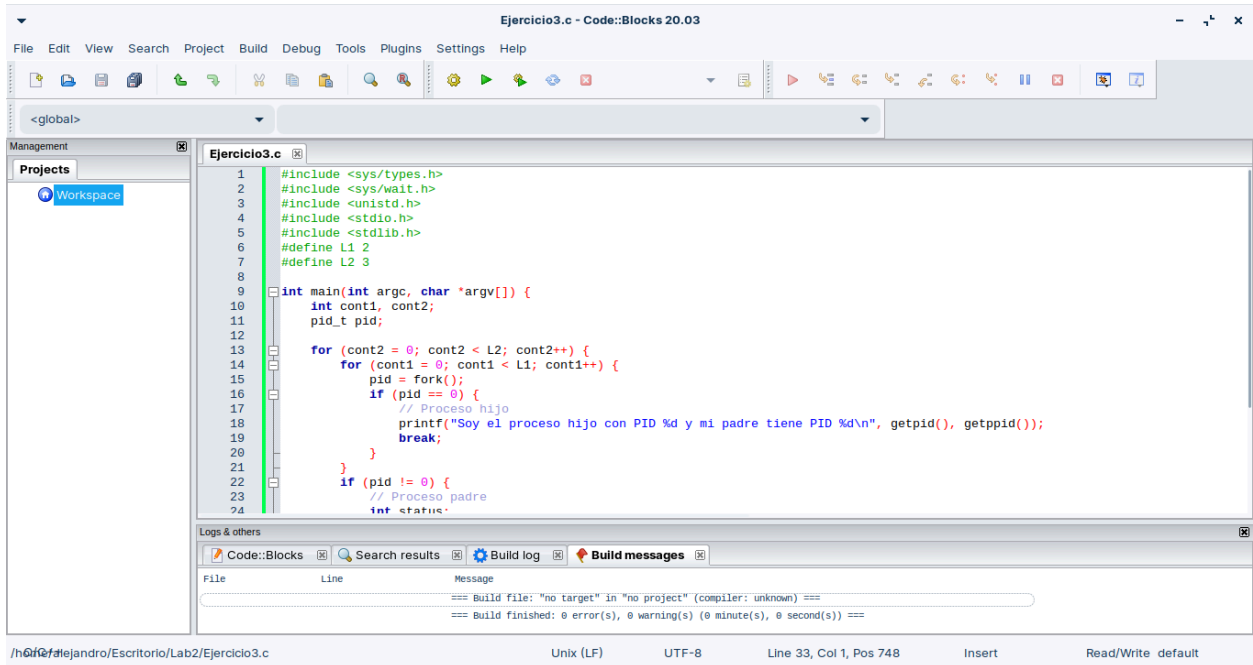
```
}return 0;
}
```

**RESPUESTA:**  
**Sin edición**



**Con la modificación:**





```

/home/alejandro/Escritorio/Lab2/Ejercicio3
Soy el proceso hijo con PID 12300 y mi padre tiene PID 12299
Soy el proceso hijo con PID 12301 y mi padre tiene PID 12299
Soy el proceso hijo con PID 12302 y mi padre tiene PID 12300
Soy el proceso hijo con PID 12304 y mi padre tiene PID 12300
Soy el proceso hijo con PID 12303 y mi padre tiene PID 12301
Soy el proceso hijo con PID 12306 y mi padre tiene PID 12304
Soy el proceso hijo con PID 12307 y mi padre tiene PID 12302
Soy el proceso hijo con PID 12308 y mi padre tiene PID 12304
Soy el proceso hijo con PID 12305 y mi padre tiene PID 12301
Soy el proceso hijo con PID 12311 y mi padre tiene PID 12303
Soy el proceso hijo con PID 12313 y mi padre tiene PID 12305
Soy el proceso hijo con PID 12310 y mi padre tiene PID 12303
Soy el proceso hijo con PID 12312 y mi padre tiene PID 12305
Soy el proceso hijo con PID 12309 y mi padre tiene PID 12302

Process returned 0 (0x0)   execution time : 0.002 s
Press ENTER to continue.

```

