



INSTITUTO POLITECNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO (ESCOM)



SISTEMAS OPERATIVOS

INTEGRANTES DEL EQUIPO:

- MARTÍNEZ GARCÍA LUIS GERARDO
- MONTESINOS GALAN URIEL
- SANTOS MÉNDEZ ULISES JESÚS

PRÁCTICA:

- ARBOL DE PROCESOS CON FORK

NÚMERO DE PRÁCTICA: 1

FECHA DE ENTREGA:

- 28/10/2021

GRUPO:

- 2CM11

Árbol de procesos con fork

Introducción: Se tiene una problemática sobre la elaboración de un árbol de procesos, el árbol tiene niveles partiendo de un proceso padre seguido de dos procesos hijos que se dividirán en rama derecha y en rama izquierda cada hijo de la rama derecha tendrá dos hijos, mientras que en la rama izquierda cada hijo principal tendrá tres hijos, este patrón se repetirá tanto para rama izquierda como para rama derecha, se busca realizar un código en C que permita realizar este árbol cumpliendo con los parámetros establecidos, los casos son:

- 1) Realizar una prueba con un nivel tres del árbol, el programa debe actuar de forma simultánea y de forma serializada.
- 2) Realizar una prueba con un nivel trece, se supone que se debe generar un error debido a que es un árbol demasiado grande, ningún proceso muere hasta llegar al nivel indicado.

Se busca que la salida en la terminal sea de una mezcla de los procesos indicando el PID de cada proceso.

Desarrollo:

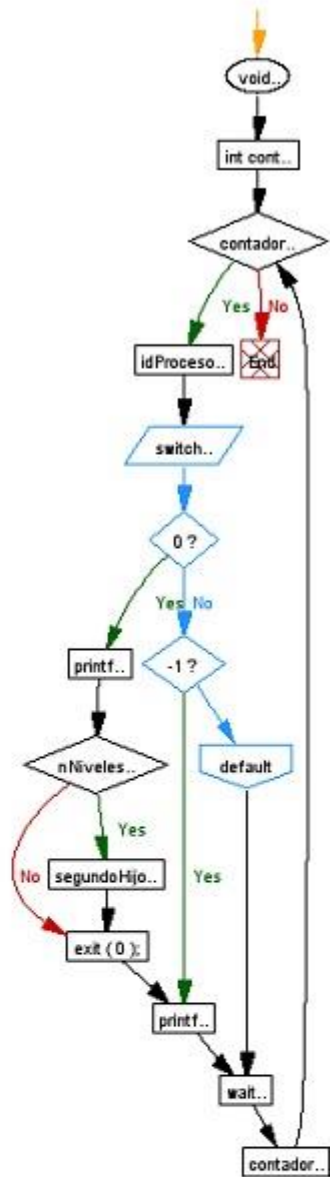
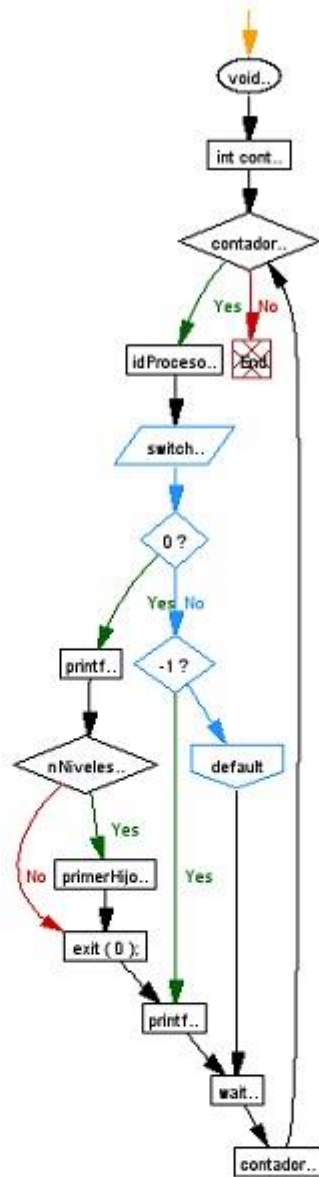
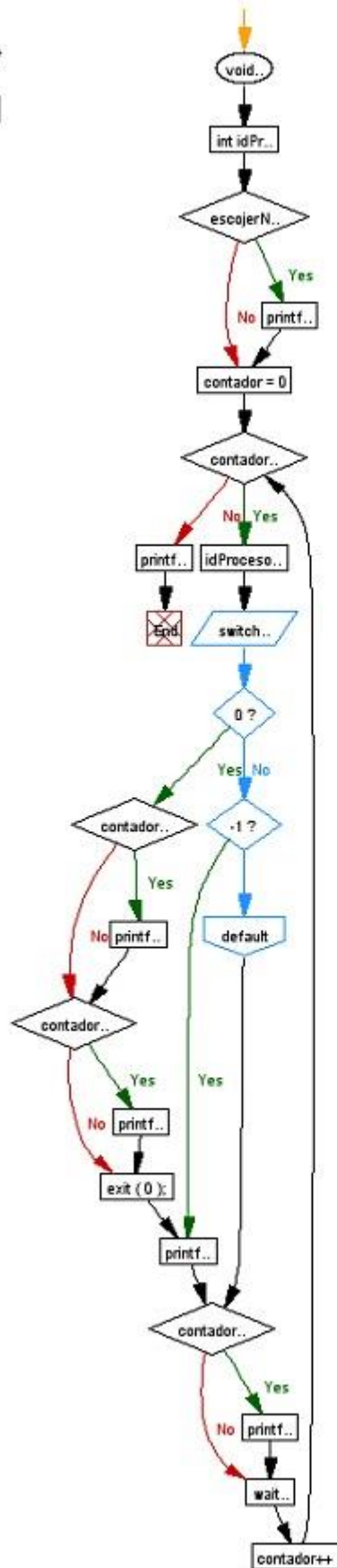
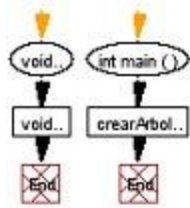
1) Diagrama de Flujo

En el siguiente diagrama podemos apreciar como las funciones main acaba de inmediato ya que esta función sólo llamará a la función crearArbol, y a su vez la función crearArbol llamará a las funciones primerHijo segundoHijo. Estas dos funciones son recursivas como lo podemos ver en el diagrama de flujo, en un determinado momento cada función se llamará a sí misma, controladas por la variable contador, que es esta variable es la que nos ayudará a asignar los niveles que el usuario ingrese.

Podemos observar, condiciones switch, que nos ayudarán a darle instrucciones específicas a los procesos padres e hijos, así como imprimir un mensaje de error, dependiendo del valor que nos devuelva la función fork().

Podemos ver algunas condiciones if en el diagrama de flujo, que nos ayudan a verificar si el hijo necesita hacer dos hijos o tres, dependiendo si pertenece a la rama izquierda o derecha, podemos observar también como todas las funciones recursivas se acercan a la condición de corte que en este caso es controlada por la variable contador, ya que en determinado momento cuando la variable nivel cumpla la condición, el programa terminará.

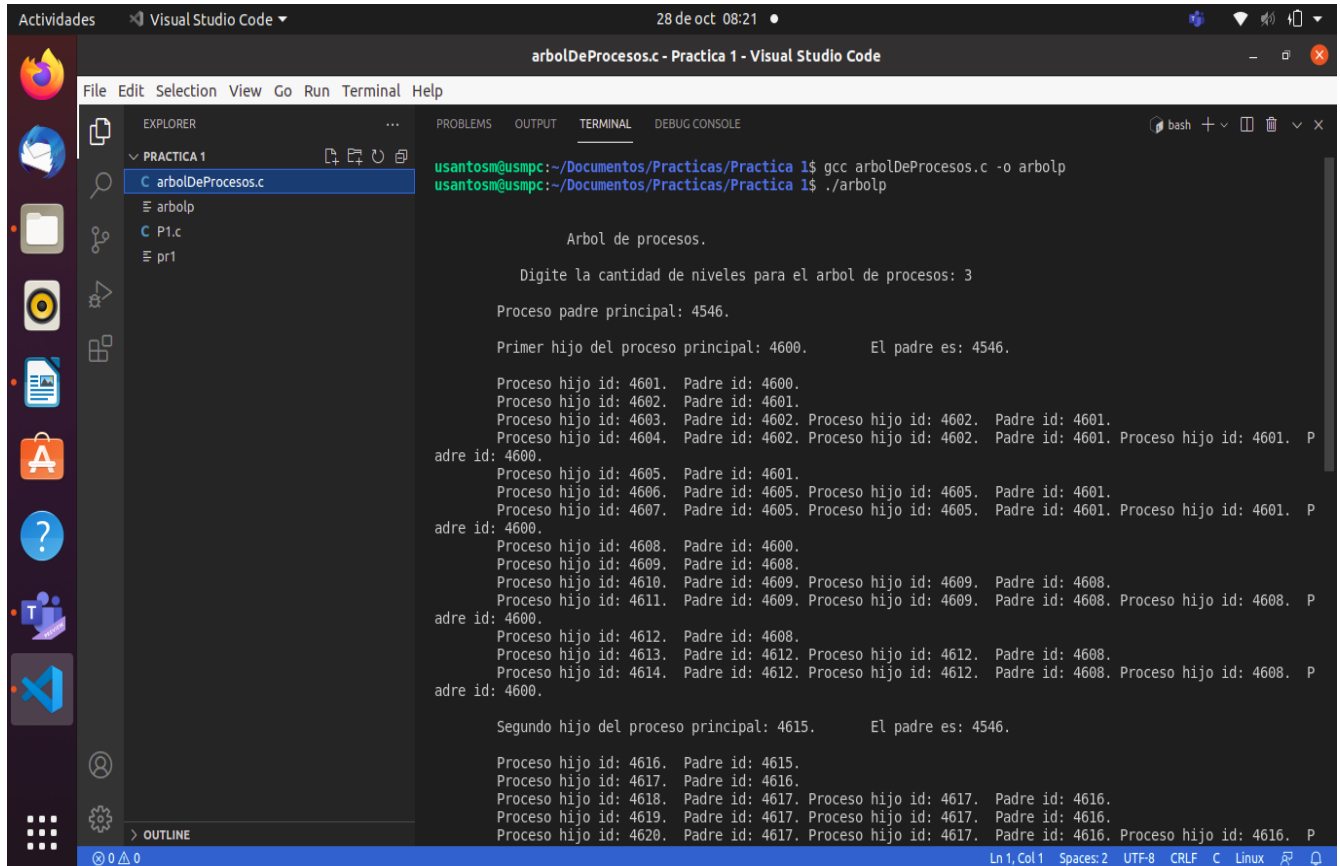
Es importante resaltar la importancia de la función wait() ya que esta nos permite que los padres esperen a que los hijos terminen.



2) Ejecución del Programa

2.1) La primera prueba consiste en ejecutar con un nivel 3 el árbol de procesos

Salida de terminal:



```
usantosm@usmpc:~/Documentos/Practicas/Practica 1$ gcc arbolDeProcesos.c -o arbolp
usantosm@usmpc:~/Documentos/Practicas/Practica 1$ ./arbolp

Arbol de procesos.

Digite la cantidad de niveles para el arbol de procesos: 3

Proceso padre principal: 4546.

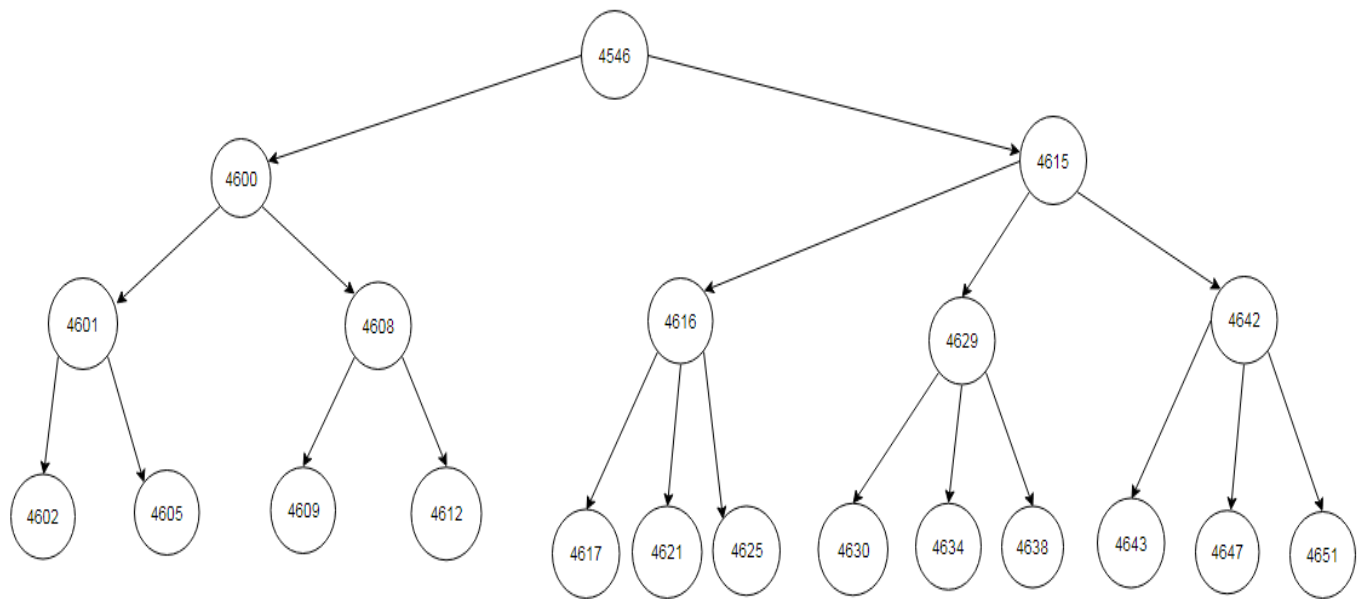
Primer hijo del proceso principal: 4600.      El padre es: 4546.

Proceso hijo id: 4601. Padre id: 4600.
Proceso hijo id: 4602. Padre id: 4601.
Proceso hijo id: 4603. Padre id: 4602. Proceso hijo id: 4602. Padre id: 4601.
Proceso hijo id: 4604. Padre id: 4602. Proceso hijo id: 4602. Padre id: 4601. Proceso hijo id: 4601. P
adre id: 4600.
Proceso hijo id: 4605. Padre id: 4601.
Proceso hijo id: 4606. Padre id: 4605. Proceso hijo id: 4605. Padre id: 4601.
Proceso hijo id: 4607. Padre id: 4605. Proceso hijo id: 4605. Padre id: 4601. Proceso hijo id: 4601. P
adre id: 4600.
Proceso hijo id: 4608. Padre id: 4600.
Proceso hijo id: 4609. Padre id: 4608.
Proceso hijo id: 4610. Padre id: 4609. Proceso hijo id: 4609. Padre id: 4608.
Proceso hijo id: 4611. Padre id: 4609. Proceso hijo id: 4609. Padre id: 4608. Proceso hijo id: 4608. P
adre id: 4600.
Proceso hijo id: 4612. Padre id: 4608.
Proceso hijo id: 4613. Padre id: 4612. Proceso hijo id: 4612. Padre id: 4608.
Proceso hijo id: 4614. Padre id: 4612. Proceso hijo id: 4612. Padre id: 4608. Proceso hijo id: 4608. P
adre id: 4600.

Segundo hijo del proceso principal: 4615.      El padre es: 4546.

Proceso hijo id: 4616. Padre id: 4615.
Proceso hijo id: 4617. Padre id: 4616.
Proceso hijo id: 4618. Padre id: 4617. Proceso hijo id: 4617. Padre id: 4616.
Proceso hijo id: 4619. Padre id: 4617. Proceso hijo id: 4617. Padre id: 4616.
Proceso hijo id: 4620. Padre id: 4617. Proceso hijo id: 4617. Padre id: 4616. Proceso hijo id: 4616. P
```

[illegible]



2.2) La segunda prueba se realizó, no obstante, las premisas fueron correctas, se produjo un error y el ciclo se hizo infinito generando así muchos procesos.

```

File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
bash

adre id: 664438.
Proceso hijo id: 664461. Padre id: 664452.
Proceso hijo id: 664462. Padre id: 664461. Proceso hijo id: 664461. Padre id: 664452.
Proceso hijo id: 664463. Padre id: 664461. Proceso hijo id: 664461. Padre id: 664452.
Proceso hijo id: 664464. Padre id: 664461. Proceso hijo id: 664461. Padre id: 664452. Proceso hijo id: 664452. P
adre id: 664438. Proceso hijo id: 664438. Padre id: 664357.
Proceso hijo id: 664465. Padre id: 664438.
Proceso hijo id: 664466. Padre id: 664465.
Proceso hijo id: 664467. Padre id: 664466. Proceso hijo id: 664466. Padre id: 664465.
Proceso hijo id: 664468. Padre id: 664466. Proceso hijo id: 664466. Padre id: 664465.
Proceso hijo id: 664469. Padre id: 664466. Proceso hijo id: 664466. Padre id: 664465. Proceso hijo id: 664465. P
adre id: 664438. Proceso hijo id: 664470. Padre id: 664465.
Proceso hijo id: 664471. Padre id: 664470. Proceso hijo id: 664470. Padre id: 664465.
Proceso hijo id: 664472. Padre id: 664470. Proceso hijo id: 664470. Padre id: 664465.
Proceso hijo id: 664473. Padre id: 664470. Proceso hijo id: 664470. Padre id: 664465. Proceso hijo id: 664465. P
adre id: 664438. Proceso hijo id: 664474. Padre id: 664465.
Proceso hijo id: 664475. Padre id: 664474. Proceso hijo id: 664474. Padre id: 664465.
Proceso hijo id: 664476. Padre id: 664474. Proceso hijo id: 664474. Padre id: 664465.
Proceso hijo id: 664477. Padre id: 664474. Proceso hijo id: 664474. Padre id: 664465. Proceso hijo id: 664465. P
adre id: 664438. Proceso hijo id: 664438. Padre id: 664357. Proceso hijo id: 664357. Padre id: 664235. Proceso hijo i
d: 664235. Padre id: 663870.
Proceso hijo id: 664478. Padre id: 664235.
Proceso hijo id: 664479. Padre id: 664478.
Proceso hijo id: 664480. Padre id: 664479.
Proceso hijo id: 664481. Padre id: 664480.
Proceso hijo id: 664482. Padre id: 664481. Proceso hijo id: 664481. Padre id: 664480.
Proceso hijo id: 664483. Padre id: 664481. Proceso hijo id: 664481. Padre id: 664480.
Proceso hijo id: 664484. Padre id: 664481. Proceso hijo id: 664481. Padre id: 664480. Proceso hijo id: 664480. P
adre id: 664479. Proceso hijo id: 664485. Padre id: 664480.
Proceso hijo id: 664486. Padre id: 664485. Proceso hijo id: 664485. Padre id: 664480.
Proceso hijo id: 664487. Padre id: 664485. Proceso hijo id: 664485. Padre id: 664480.
Proceso hijo id: 664488. Padre id: 664485. Proceso hijo id: 664485. Padre id: 664480. Proceso hijo id: 664480. P
adre id: 664479. Proceso hijo id: 664489. Padre id: 664480.
Proceso hijo id: 664490. Padre id: 664489. Proceso hijo id: 664489. Padre id: 664480.
  
```



```
Proceso hijo id: 665273.    Padre id: 665271.    Proceso hijo id: 665271.    Padre id: 665262.
Proceso hijo id: 665274.    Padre id: 665271.    Proceso hijo id: 665271.    Padre id: 665262.    Proceso hijo id: 665262.    P
adre id: 665248.    Proceso hijo id: 665248.    Padre id: 665207.
Proceso hijo id: 665275.    Padre id: 665248.
Proceso hijo id: 665276.    Padre id: 665275.
Proceso hijo id: 665277.    Padre id: 665276.    Proceso hijo id: 665276.    Padre id: 665275.
Proceso hijo id: 665278.    Padre id: 665276.    Proceso hijo id: 665276.    Padre id: 665275.
Proceso hijo id: 665279.    Padre id: 665276.    Proceso hijo id: 665276.    Padre id: 665275.    Proceso hijo id: 665275.    P
adre id: 665248.
Proceso hijo id: 665280.    Padre id: 665275.
Proceso hijo id: 665281.    Padre id: 665280.    Proceso hijo id: 665280.    Padre id: 665275.
Proceso hijo id: 665282.    Padre id: 665280.    Proceso hijo id: 665280.    Padre id: 665275.
Proceso hijo id: 665283.    Padre id: 665280.    Proceso hijo id: 665280.    Padre id: 665275.    Proceso hijo id: 665275.    P
adre id: 665248.
Proceso hijo id: 665284.    Padre id: 665275.
Proceso hijo id: 665285.    Padre id: 665284.    Proceso hijo id: 665284.    Padre id: 665275.
Proceso hijo id: 665286.    Padre id: 665284.    Proceso hijo id: 665284.    Padre id: 665275.
Proceso hijo id: 665287.    Padre id: 665284.    Proceso hijo id: 665284.    Padre id: 665275.    Proceso hijo id: 665275.    P
adre id: 665248.    Proceso hijo id: 665248.    Padre id: 665207.    Proceso hijo id: 665207.    Padre id: 664964.
Proceso hijo id: 665288.    Padre id: 665207.
Proceso hijo id: 665289.    Padre id: 665288.
Proceso hijo id: 665290.    Padre id: 665289.
Proceso hijo id: 665291.    Padre id: 665290.    Proceso hijo id: 665290.    Padre id: 665289.
Proceso hijo id: 665292.    Padre id: 665290.    Proceso hijo id: 665290.    Padre id: 665289.
Proceso hijo id: 665293.    Padre id: 665290.    Proceso hijo id: 665290.    Padre id: 665289.    Proceso hijo id: 665289.    P
adre id: 665288.
Proceso hijo id: 665294.    Padre id: 665289.
Proceso hijo id: 665295.    Padre id: 665294.    Proceso hijo id: 665294.    Padre id: 665289.
Proceso hijo id: 665296.    Padre id: 665294.    Proceso hijo id: 665294.    Padre id: 665289.
Proceso hijo id: 665297.    Padre id: 665294.    Proceso hijo id: 665294.    Padre id: 665289.    Proceso hijo id: 665289.    P
adre id: 665288.
Proceso hijo id: 665298.    Padre id: 665289.
Proceso hijo id: 665299.    Padre id: 665298.    Proceso hijo id: 665298.    Padre id: 665289.
Proceso hijo id: 665300.    Padre id: 665298.    Proceso hijo id: 665298.    Padre id: 665289.
Proceso hijo id: 665301.    Padre id: 665298.    Proceso hijo id: 665298.    Padre id: 665289.    Proceso hijo id: 665289.    P
adre id: 665288.    Proceso hijo id: 665288.    Padre id: 665207.
^C
usantosm@usmpc:~/Documentos/Practicas/Practica 1$
```

Debido a un problema con la función recursiva de procesos se hizo infinito generando así un error en el programa.

Conclusiones:

Martínez García Luis Gerardo: En conclusión, la realización de esta práctica nos permite ver cómo es que trabaja el sistema operativo a la hora de crear procesos, ya que realizando de manera correcta dicha práctica, podemos observar que el sistema operativo tiene un límite para la creación de procesos simultáneos, haciendo que después de una determinada cantidad empiece a generar conflictos.

Montesinos Galan Uri: Con esta práctica, me quedo mucho más claro el funcionamiento de la función fork, lo mejor fue que todas las dudas que tenía con esta función fueron aclaradas con esta práctica, pude comprender los límites y como un árbol de procesos puede crecer exponencialmente, estoy seguro de que esto será de gran ayuda en mi formación profesional.

Santos Méndez Ulises Jesús: En conclusión, la práctica fue un tanto laboriosa, ya que hubo conflicto con la realización de la iteración para la elaboración de los primeros procesos y que se hicieran simultáneos a su vez también lo fue con las funciones recursivas para los hijos ya que hubo problemas con la creación de los hijos ya que o podían ser más o menos, al final se obtuvo un resultado esperado.