

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO
SISTEMAS OPERATIVOS
GRUPO: 2CM8

ALUMNO:
NAVA VIVAS ANA PAOLA

PRACTICA 2: FORK

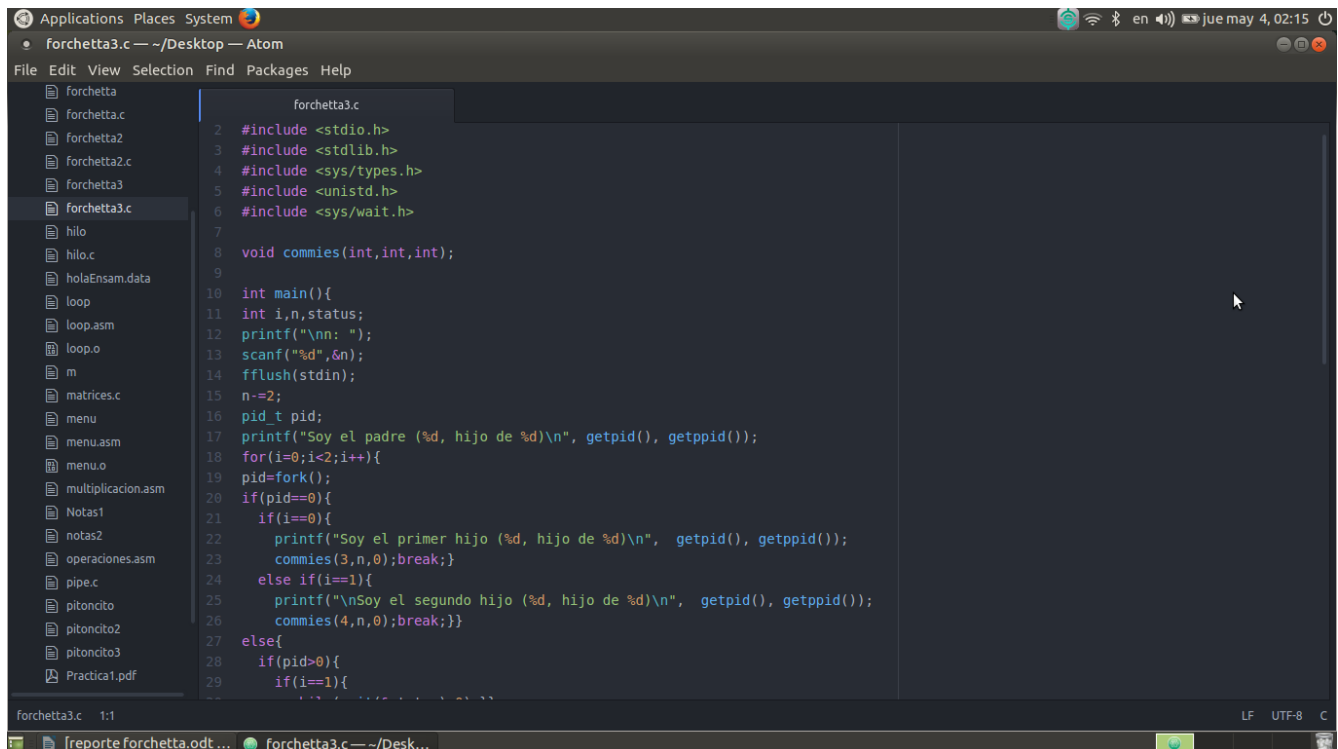
La práctica consistió en realizar un árbol de n generaciones, donde un padre tuviera un hijo derecho y un hijo izquierdo, pero a partir de la segunda generación, los hijos izquierdos tuvieran 3 hijos y los hijos derechos, 4 hijos.

Primero declaré el prototipo de función `commies`, que recibiría 3 enteros.

En el `main` se pide el nivel que tendrá el árbol, el usuario lo introduce y se le resta 2, porque a la función no le interesa la generación del padre ni la primera descendencia.

Se captura el proceso del padre con el primer `getpid`, `getppid` captura el proceso de la terminal que es el padre del padre inicial.

Dentro de un `for` que itera dos veces, se obtienen los primero dos hijos, y de paso se introducen los valores a la función `commies`.



```
forchetta3.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/types.h>
4  #include <unistd.h>
5  #include <sys/wait.h>
6
7  void commies(int,int,int);
8
9
10 int main(){
11     int i,n,status;
12     printf("\nn: ");
13     scanf("%d",&n);
14     fflush(stdin);
15     n-=2;
16     pid_t pid;
17     printf("Soy el padre (%d, hijo de %d)\n", getpid(), getppid());
18     for(i=0;i<2;i++){
19         pid=fork();
20         if(pid==0){
21             if(i==0){
22                 printf("Soy el primer hijo (%d, hijo de %d)\n", getpid(), getppid());
23                 commies(3,n,0);break;}
24             else if(i==1){
25                 printf("\nSoy el segundo hijo (%d, hijo de %d)\n", getpid(), getppid());
26                 commies(4,n,0);break;}}
27         else{
28             if(pid>0){
29                 if(i==1){
```

En la función `commies`, dependerá de si se trata del hijo izquierdo o derecho, si es el izquierdo se tendrán los argumentos (3,n,0), es decir: 3 hijos, n niveles, y 0 en contador. Si es para el hijo derecho se tendrá (4,n,0).

```
forchetta3.c
30 while(wait(&status)>0);}}
31 }}
32 }
33
34 void commies(int d,int n,int cont){//d=derecho o izquierdo, n=nivel, contador...
35 pid_t pid;
36 int i,status;
37 if(cont==n){exit(0);}
38 for(i=0;i<d;i++){
39 pid=fork();
40 if(pid==0){
41 printf("\nSoy %d, hijo de %d\n", getpid(), getppid());
42 commies(d,n,cont+1);break;}
43 else{
44 if(pid>0){
45 if(i==d-1){
46 while(wait(&status)>0);}
47 }}
48 }}
49
50 /*
51 nivel =0
52 for(i=0,i<2,i++){
53 id=fork()
54 id==0{
55 if(i==0){
56 nivel++
57 recursiva(nivel,stop,3)}
58 }
```

La función es recursiva, empieza obteniendo el proceso actual y mediante un proceso similar al caso del primer y segundo hijo del padre original, se obtendrán 3 o 4 hijos, todo depende.

La función se detiene cuando el contador llega al último nivel.

Conclusiones

En esta práctica aprendí que incluso el proceso padre, tiene un padre, y es la terminal. Me quedó claro el uso de `getpid` y `getppid`, que sirven para obtener el proceso que esta corriendo y el padre de este respectivamente, aprendí que la función `exit(0)` sirve para matar un proceso y por tanto la utilicé para salir de la función recursiva.

La ejecución es un poco pesada, porque cada hijo copia el proceso del padre, entonces cada proceso tiene todo el programa copiado en el espacio de memoria de su proceso, es por eso que el programa muere en la septima u octava generación, ya que hace crecer el stack de manera muy rápida.

Práctica
árbol
procesos

Ana Paola ~~Alvar~~ Vivas

