Practica No. 2
Creación de un árbol de procesos

Sistemas Operativos

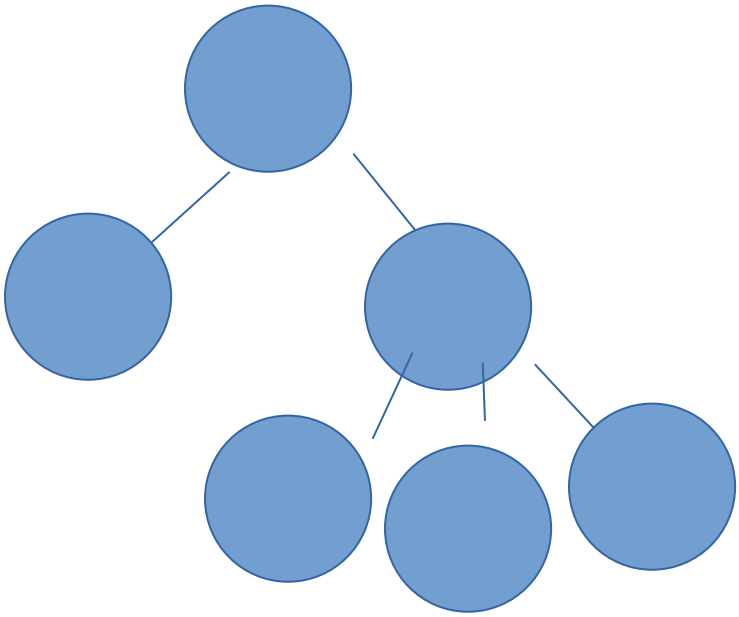
Josue Rangel Gonzales

Grupo: 2CV7

Barón Hernández Diego Ismael

Introducción

Una situación muy habitual dentro de un programa es la de crear un nuevo proceso que se encargue de una tarea concreta, descargando al proceso principal de tareas secundarias que pueden realizarse asíncronamente o en paralelo. Linux ofrece varias funciones para realizar esto: system(), fork() y exec().
Bien para esta practica se utilizara el metodo que utiliza fork() para la creacion de un arbol de procesos en el que el primer hijo derecho tendra 3 hijos mientras que el primer hijo izquierdo tendra 4 hijos, y asi sucesivamente para cada respectiva rama.



El objetivo es que el usuario especifique el nivel hasta el que quiere llegar y esto desencadenara que el arbol crezca exponencialmente.

Desarrollo
Lo primero que se hizo fue poder realizar la creación de n procesos, lo cual se logro con el siguiente codigo:

```c
for (i = 1; i < 3; ++i)
{
    pid_t pid;
    int estado;

    pid = fork();
    switch(pid)
    {
        case -1: /*error del fork*/
            perror("Error en el fork");
            break;
        case 0: /*proceso hijo*/
            if (i == 1)
            { ...
            }
            if (i == 2)
            { ...
            }
            break;
        default:
            /*proceso padre*/
            if (i == 1)
            {
                printf("Soy el padre: PID %d; PPID = %d\n", getpid(), getppid());
            }
            wait(&estado);
    }
}
```

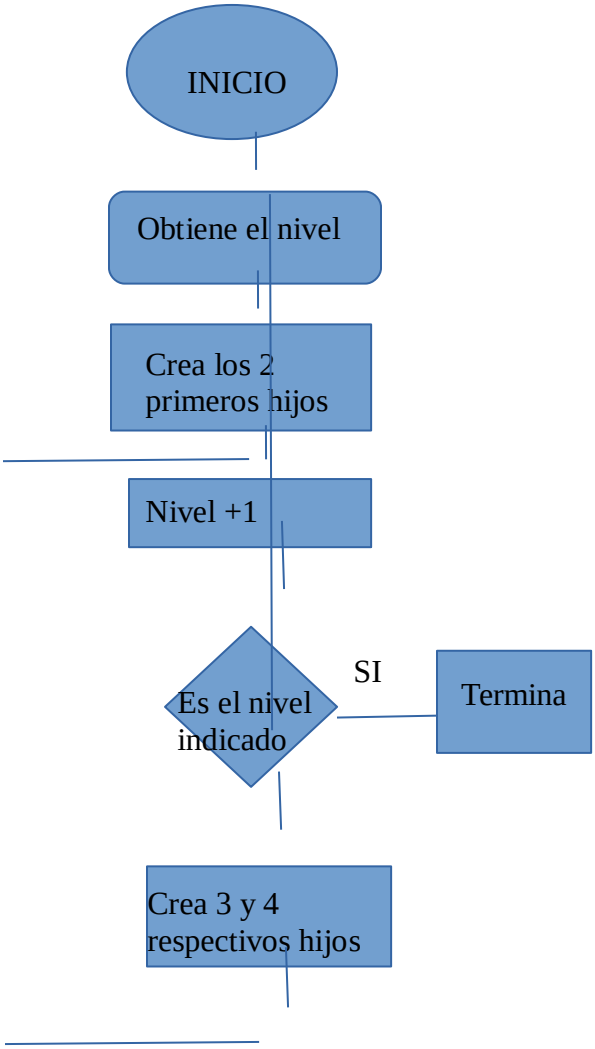
Lo siguiente fue poder crear para cada hijo la cantidad requerida de hijos lo cual se consiguio añadiendo dos if para los casos en que se crean cada uno de los primeros hijos:

```
for (i = 1; i < 3; ++i)
{
    pid_t pid;
    int estado;

    pid = fork();
    switch(pid)
    {
        case -1: /*error del fork*/
            perror("Error en el fork");
            break;
        case 0: /*proceso hijo*/
            if (i == 1)
            {
                for (int j = 1; j < 5; ++j)
                {
                    pid_t pid2;
                    int estado2;

                    pid2 = fork();
                    switch(pid2)
                    {
                        case -1:
                            perror("Error en el fork");
                            break;
                        case 0:
                            printf("Soy el hijo derecho %d: PID %d; PPID = %d\n", j, getpid(
                            exit(0);
                            break;
                        default:
                            if (j == 1)
                            {
                                printf("Soy el hijo %d: PID %d; PPID = %d\n", i, getpid(), g
                            }
                            wait(&estado2);
                            if (j == 4)
                            {
                                exit(0);
                            }
                    }
                }
            }
```

Diagrama:


INICIO
Obtiene el nivel
Crea los 2 primeros hijos
Nivel +1
Es el nivel indicado
SI
Termina
Crea 3 y 4 respectivos hijos

Conclusiones

Se presentaron varios problemas sobretodo con el razonamiento del problema ya que para hacerlo para n cantidad de niveles, asi como la condicion de esperar de hijos ya que en un principio se serializaba el proceso de creación y no hacia paralelamente la creacion de procesos, ademas aprendi que al llegar a cierto numero de procesos creados el programa ya no puede crear mas por que se alcanza el limite y arroja un error de creación en el fork, siendo como condicion de reanudacion que los hijos anteriores terminen.