Ejercicio de Sistemas Operativos
Parte Práctica

Problema 1

Dado el siguiente programa:

```
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <stdlib.h>
4
5  int main()
6  {
7     int id1, id2, id3;
8
9     id1 = fork();
10
11    if (id1 == 0)
12    {
13       sleep(4);
14       printf("Orden␣1:␣%d␣%d\n", getpid(), getppid());
15       exit(0);
16    }
17
18    else
19    {
20       id2 = fork();
21       if ( id2 == 0)
22       {
23          id3 = fork();
24          if (id3 == 0)
25          {
26             sleep(6);
27             printf("Orden␣2:␣%d␣%d\n", getpid(), getppid());
28             exit(0);
29          }
30          else
31          {
32             printf("Orden␣3:␣%d␣%d\n", getpid(), getppid());
33             sleep(10);
34             exit(0);
35          }
36       }
37       else
38       {
39          printf("Orden␣4:␣%d␣%d\n", getpid(), getppid());
40          exit(0);
41       }
42     }
43  }
```

Responda a las preguntas que a continuación se plantean relativas a su ejecución:

1. Represente un esquema de la jerarquía de procesos que se crea cuando se ejecuta el programa anterior. Indique la relación de parentesco entre procesos de la forma que se indica en la figura 1, y especifique al lado de cada proceso la línea de código cuya ejecución lo crea.

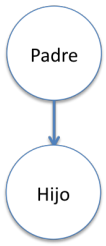

Figura 2: Representación de la relación entre proceso padre y proceso hijo.

RESPUESTA:

2. Suponiendo que el PID de la shell que ejecuta el programa es 1967, que no se crean más procesos que los implicados en la ejecución del programa del enunciado, que los PIDs en

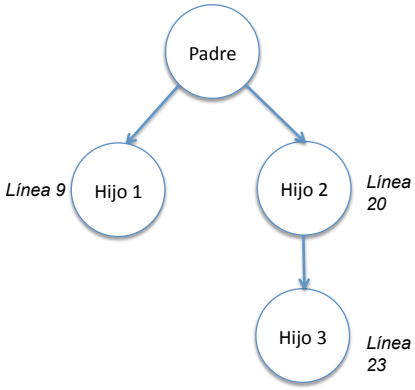

Figura 3: Jerarquía de procesos.

este sistema se asignan correlativamente, y que el primer PID disponible es 2890, indique cuál sería la salida por pantalla al ejecutarse el programa anterior[2]

RESPUESTA: Suponiendo que el tiempo de ejecución de las llamadas a funciones (excepto las invocaciones al servicio POSIX `sleep()`) es despreciable, las líneas que se obtendrían por pantalla serían las siguientes:
Orden 4: 2890 1967
Orden 3: 2892 1
Orden 1: 2891 1
Orden 2: 2893 2892

3. Si añadiéramos en el código del programa una variable global `i` inicializada a `0` y sustituyéramos cada línea `printf()` por las líneas siguientes:

```
i=i+1;
printf(''Orden %d: %d %d'', i, getpid(), getppid())
```

¿Qué valores de `i` saldrían en pantalla? Razone la respuesta.

RESPUESTA: Cada proceso que se crea tiene una copia propia del área de datos globales del proceso padre. Por lo tanto, el cambio realizado sólo originaría líneas cuyo valor de i sería siempre 1, que no es la secuencia que se generaría con la alternativa del código inicial

4. Se ha realizado independientemente un programa denominado `MapaDeMemoria` que se encarga de visualizar en pantalla el mapa de memoria actual del proceso que lo invoca. La forma de ejecutar este programa es la siguiente:

```
$ MapaDeMemoria
```

Si el proceso que ejecuta la línea 14 se tuviera que encargar, además, de ejecutar el programa `MapaDeMemoria`, ¿qué nuevas instrucciones habría que añadir al código existente y en qué lugar? Razone la respuesta.

RESPUESTA: Si tiene que ejecutar el nuevo programa, tenemos que utilizar cualquier servicio POSIX de la familia `exec`, por ejemplo, `execl`. Además, se

[2]NOTAS:

- En UNIX, si un proceso padre finaliza antes que sus procesos hijos, estos procesos son adoptados por el proceso *init*; a partir de ese instante, *init* es el nuevo proceso padre de estos procesos hijos.
- Los servicios POSIX `getpid()` y `getppid()` devuelven el identificador del proceso, y el identificador del proceso padre respectivamente.

debe invocar tras la línea 14, porque después no se ejecutarán más sentencias del código de este programa. Por lo tanto, en la línea 15 se añadiría lo siguiente:

```
execl("./MapaDeMemoria", "./MapaDeMemoria", NULL);
```

5. ¿Se podría garantizar con el uso de las llamadas `wait`, `exit`, `fork` y `exec` que el orden de aparición de las líneas en pantalla fuera secuencial y ascendente (Orden1, Orden2, Orden3, etc.)? Razone la respuesta y, en caso afirmativo, indique cómo.

   RESPUESTA: Sí, ejecutando el proceso padre en la línea 39, antes de la invocación a `printf()`:
   `wait(NULL);`
   `wait(NULL);`
   De este modo, el proceso padre, que imprime `Orden 4`, espera a sus dos procesos hijos y, además, para que el proceso creado al ejecutarse la línea 20 esperase, a su vez, a su proceso hijo, se añadiría en la línea 32, antes de la invocación a `printf()`, la siguiente línea:
   `wait(NULL);`