Hernández Verde Mariana
Ortiz Herrera Jesús Julian
Torres Arrendondo Evelyn Ximena
Trujillo Martinez Omar Noé

PROGRAMACIÓN AVANZADA
2TM2

Procesos y sentencia "fork()"

1.¿Cuál es la función principal de la sentencia fork()?
La función principal de la sentencia `fork()` en programación es crear un nuevo proceso duplicando el proceso existente. Esto significa que después de llamar a `fork()`, hay dos procesos ejecutándose, ambos con el mismo código y estado, pero con identificadores de proceso (PID) diferentes. El proceso que llama a `fork()` se denomina proceso padre, y el nuevo proceso creado se llama proceso hijo.

2.¿Qué librería en lenguaje C contiene la función fork()?
La función fork() está unistd.h en el lenguaje C fork(), generalmente se incluye **#include <unistd.h>**.

3. En el contexto de procesos, la sentencia fork(). ¿Crea un proceso padre o un proceso hijo?
La llamada a fork() genera una copia del proceso que la realiza, creando así dos procesos distintos: el proceso padre y el proceso hijo. Después de llamar a fork(), ambos procesos (el padre y el hijo) continúan ejecutándose a partir del punto en el que se realizó la llamada. La principal diferencia entre ellos radica en el valor que devuelve fork(). En el proceso padre, fork() devuelve el ID del proceso hijo recién creado, mientras que en el proceso hijo, devuelve 0. De esta manera, el código puede tomar decisiones basadas en el valor devuelto para ejecutar acciones específicas fork() en el proceso padre o en el hijo.

La función fork() crea un nuevo proceso que es una copia exacta del proceso que lo llamó, y ambos procesos (el padre y el hijo) continúan su ejecución a partir del punto de la llamada a fork().

4. ¿Cuál es la sintaxis (componentes) de la función fork()?
Sintaxis:

```
#include <unistd.h>

pid_t fork(void);
```

- ***pid_t*** es un tipo de datos utilizado para representar un ID de proceso. Este es un tipo entero que puede contener un identificador de proceso (PID).
- ***fork()*** devuelve el PID del proceso hijo en el proceso padre y 0 en el proceso hijo. fork() devuelve -1.

Si ocurre un error al crear un nuevo proceso. Entonces, después de llamar a fork(), puedes usar el valor de retorno para determinar qué proceso está ejecutando el código y tomar decisiones basadas en eso.

5. Si tiene un programa cualquiera, con sentencias para ejecutar una tarea específica y se coloca la sentencia fork() en alguna línea del programa. ¿Qué salida tendrá, se programa?
La salida especifica que tendrá el programa que contiene fork() depende del flujo de ejecución ejecución ya que después de esta se dividen en dos procesos los cuales se ejecutan a partir de la llamada fork() y su valor de retorno del padre e hijo será el valor de retorno de fork().

6. ¿Qué es el PID y que tipo de dato es?
Es una variable que se utiliza comúnmente en programación en sistemas operativos tipo Unix para representar el ID de un proceso. La abreviatura "pid" proviene de "Process ID" en inglés, que se traduce como "Identificador de Proceso".

7. Si en un programa se manda a llamar a la función fork() 2 veces antes de alguna otra sentencia. ¿Cuántas veces se ejecuta la salida del programa?
La función fork() crea dos procesos y los ejecuta: el proceso padre y el proceso hijo.
Si se llama a la función fork() dos veces antes de alguna sentencia específica en un programa, se crearán dos procesos en cada llamada, dando lugar a un total de 4 procesos en el momento de esa sentencia. Esto se debe a que cada llamada a fork() duplica el proceso que la llama.

En cada proceso, la salida del programa se ejecutará independientemente. Por lo tanto, si hay una sentencia de salida después de las dos llamadas a fork(), se ejecutará cuatro veces, una vez por cada proceso creado.