7

Practica 2

Sistemas Operativos

Universidad Autónoma Metropolitana – Azcapotzalco

1 de junio de 2018

Índice

1. Introducción

Un proceso se puede decir que es un programa en ejecución, es necesario ampliar este concepto para poder distinguir entre un programa y un proceso ya que hay grandes diferencias entre estos dos. El primero se refiere en gran medida a una entidad pasiva que esta compuesta por una lista de instrucciones que se encuentran almacenadas en memoria (lo cual se conoce como archivo ejecutable) mientras que el segundo es una entidad activa ya que las instrucciones asociadas a este ya están cargadas en memoria y por consiguiente el procesador podría estar por ejecutarlas. El estado de un proceso se define por la actividad que este se encuentre realizando, existen tres estados en los que este se puede encontrar, los cuales son Ejecución, Bloqueado y Listo. Los procesos transitan en estos estados durante su ejecución y su transición´se debe a señales o eventos externos y al planificador de procesos.

Una llamada al sistema[1] proporciona los servicios que el sistema operativo ofrece, como lo es el acceso a memoria o a los dispositivos de E/S. Un programa, por mas sencillo que sea, requerirá varias llamadas al sistema y estas requieren tener un conocimiento más detallado sobre el sistema operativo que se use por lo que su implementación no es sencilla para muchos desarrolladores, es por esta razón que la mayoría de desarrolladores de aplicación utilizan una API (Appllication Programming Interface) la cual simplifica en gran medida el uso de llamadas al sistema pues especifican un conjunto de funciones en donde según sea el caso se pueden indicar los parámetros que puede recibir y los valores de retorno que devuelven. Una de de estas API que principalmente fue desarrollada para sistemas UNIX es POSIX (Portable Operating System Interface) la cual incluye una serie de comandos y programas de utilidades escritos por la IEEE. Un tipo de llamadas al sistema es el control de procesos, donde estas proporcionan las funciones necesarias para la gestión de los procesos como lo son terminar, crear, ejecutar, entre otras.

2. Desarrollo

Para crear un nuevo proceso a partir de uno ya existente se usa la función "fork", la cual se incluye en la biblioteca ünistd.h", esta función regresara el id del proceso creado. La practica consiste en crear una serie de estructuras de procesos a partir de esta función. Se debe tomar en cuenta que al crear el nuevo proceso, el proceso padre seguirá ejecutando su próxima instrucción y tendrá el id de su hijo en la variable donde se haya llamado a fork, mientras que para el proceso hijo guarda el estado del padre al momento que este lo haya creado, es decir, la próxima instrucción que ejecutará será la misma que el padre ejecute después de la llamada a fork además de que la variable donde se llama a esta función es inicializada a cero y por consiguiente esta es una característica importante para distinguir a ambos procesos.

Para cada una de las estructuras que se realizaron se partió del diferencia anteriormente descrita entre el proceso padre y el hijo por lo que dependiendo de estas se condicionaba uno u otro para que creara un nuevo proceso y así poder construir la estructura de procesos. En las Figuras 1, 2 y 3 se encuentran el árbol de procesos del programa que se ejecutó para obtener cada una de las estructuras,en estas se puede ver claramente que se cumplen con las estructuras requeridas.

*No se pidió esta estructura

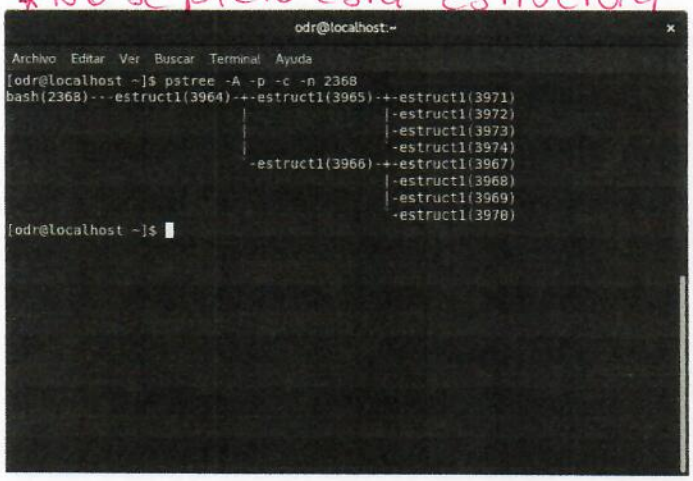

Figura 1: Estructura 1.

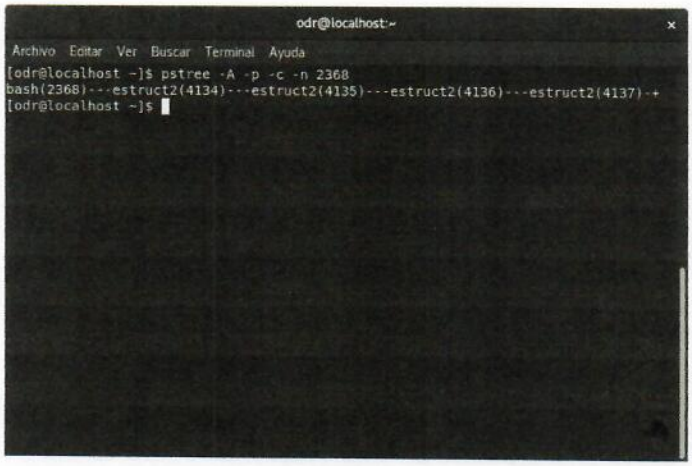

Figura 2: Estructura 2.

*

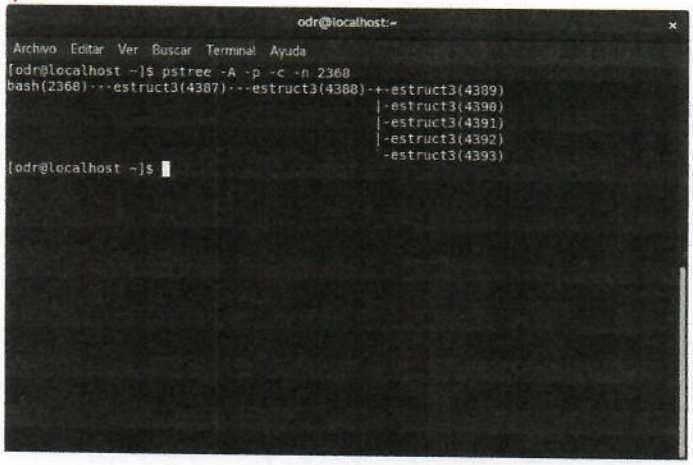

Figura 3: Estructura 3.

3. Cuestionario

1.- Se crearon procesos mediante fork donde estos podían depender de condiciones o no para tener la estructura que se requería. 1/2

2.- En la primera estructura y en las demás se llamo a la función sleep para que el programa se terminara en 10 segundos al llamar a esta función y así poder esperar que el hijo terminara.

3.- Las llamadas al sistema utilizadas hasta ahora son las de control de procesos (fork, sleep) y administración de archivos (write).

4.- Para un proceso padre espere a que su proceso hijo termine se puede usar la función "wait"[2] la cual pone al proceso en espera hasta que el proceso hijo cambie de estado.

4. Conclusión

El uso de una API como lo es POSIX simplifica mucho el uso de llamadas al sistemas que se realizan en el diseño de un programa pues incluyen funciones ya definidas por lo que acciones como la creación de un nuevo proceso se puede realizar de una más intuitiva aun sin conocer a fondo lo que el sistema operativo realiza para esto. La gestión de estos procesos es importante ya que podemos hacer que los procesos se creen a partir de condiciones que podemos establecer dando lugar a que se puedan hacer estructuras de procesos como las que se muestran en el desarrollo de la practica, sin embargo, el uso de la función "sleep"para hacer que los procesos padre esperen a que sus hijos ejecuten su programa pudiera ser no muy conveniente en algunos casos pues el tiempo va a variar dependiendo de la cantidad de procesos que se tengan y las instrucciones que estos realicen por lo que el uso de la función "wait"deberá ser más conveniente en casos como estos.

Referencias

1. . G. G. Silberschatz A., Galvin P., *Fundamentos de Sistemas Operativos*, 7.ª ed. McGraw-Hill, 2006, pp. 39–43.
2. *fork - Create a new process*, [Web; accedido el 31-05-2018]. [En línea]. Disponible: URL{https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.bpxbd00/rtfor.htm}