PROCESOS

Un proceso es una entidad abstracta, definida por el kernel, para que los recursos del sistema se localicen de modo que la CPU ejecute un programa es decir una instancia de un programa en ejecución y el estado de los registros. Los procesos se ejecutan en pseudo paralelo (pseudoparalelismo es para un solo procesador). Debido a esto aprovecho tiempos muertos y hago posible que demore menos la ejecución de tareas haciendo rápida la conmutación de la CPU esto se define como multiprogramación.

¿Para qué quiero más de un programa ejecutándose?, es dado que la velocidad del procesador es mayor comparada a la velocidad de entrada/salida.

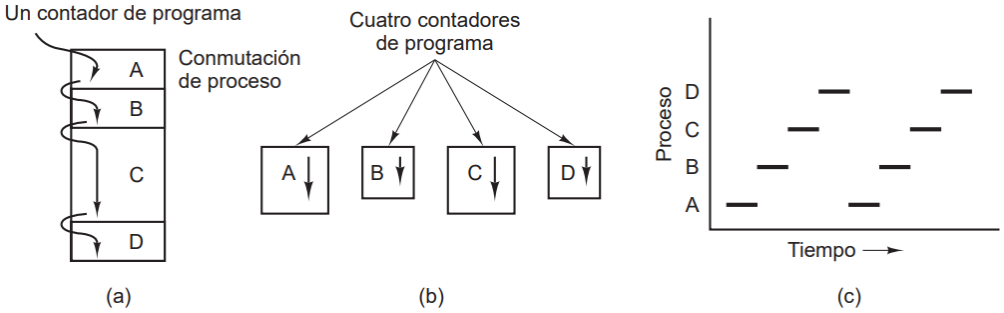


Figura 2-1. (a) Multiprogramación de cuatro programas. (b) Modelo conceptual de cuatro procesos secuenciales independientes. (c) Sólo hay un programa activo a la vez.

Los programas deben estar en memoria para que se ejecuten. Si hacemos cosas de tiempo real, esto no es recomendable porque no puedo basarme en supuestos de tiempo, dado que los tiempos no siempre son exactos.

¿Qué es un proceso?

Un Proceso (entidad activa) es lo que se está ejecutando, pero con todo su contexto: un programa (que es una entidad pasiva) en ejecución, además tiene el contador de programa, el estado de todos los registros, saber la pila de ese proceso, etc.

Varios procesos pueden estar en distintas partes de mis programas.

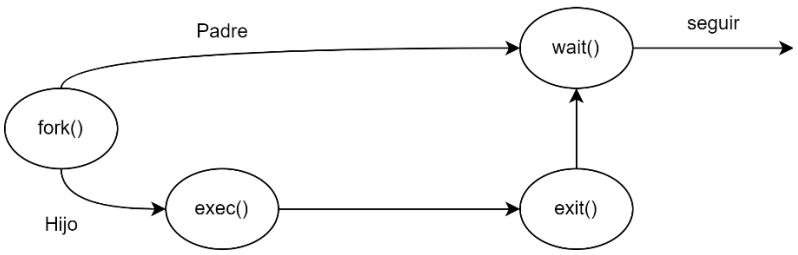
Creación de un Proceso

La creación de procesos me sirve por ejemplo para la inicialización del sistema, cada vez que inicio mi sistema se crean varios procesos denominados servicios (esto me realiza algunas tareas), pueden ser procesos de primer plano o segundo plano.

Tenemos solo una llamada a sistemas para crear un proceso esta es fork(), indica al sistema operativo que cree un proceso y le indica cual programa debe ejecutarlo. Su función es crear un clon exacto del proceso que hizo la llamada. En general el proceso hijo ejecuta después a execve o una llamada al sistema similar para cambiar su imagen de memoria y ejecutar un nuevo programa. *EL HIJO ME SIRVE PARA REALIZAR UNA TAREA ESPECIFICA.*

Creación de Procesos con fork()

Pueden existir varias alternativas a la creación de este proceso, que son las siguientes:



Alternativas Padre:

- Primera alternativa, es esperar que el Hijo termine
- Segunda alternativa, seguir ejecutándose concurrentemente con hijo (ej. servidor web)

Alternativas Hijo:

- Usa mismo programa y datos que heredo el padre
- Carga un nuevo programa

Terminación de un Proceso

Tenemos las siguientes formas de terminar un proceso:

1. Salida normal (Voluntaria)
2. Salida por error (Voluntaria)
3. Error fatal (Involuntaria)
4. Eliminado por otro proceso (Involuntaria)

Cuando un compilador ha compilado el programa que recibe, ejecuta una llamada al sistema para indicar al sistema operativo que ha terminado, esta llamada es exit. La cuarta razón por la que un proceso podría terminar es que ejecute una llamada al sistema que indique al sistema operativo que elimine otros procesos. En UNIX esta llamada es kill.

Jerarquía de Procesos

El proceso hijo puede crear por sí mismo más procesos, formando una jerarquía de procesos.

Estados de un Proceso

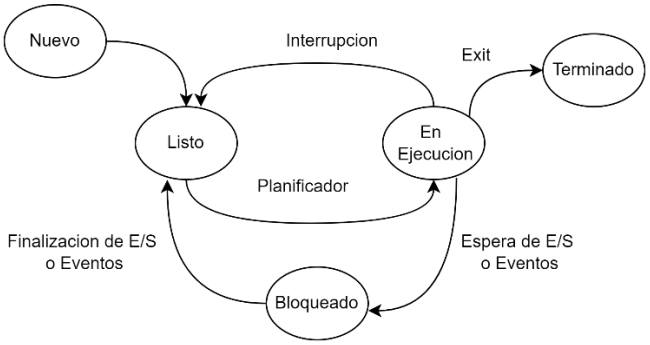
Nuevo: Cuando un proceso arranca (es nuevo) queda en una lista de los procesos que están listos para ejecutarse.

Listo a Ejecución: Una rutina del sistema operativo se va a estar corriendo en el procesador que es el planificador que es parte del programa, lo que hace es elegir los procesos que están listos y va a cambiar en la tabla de procesos el estado de listo a ejecución y automáticamente hago un cambio de tareas usando task gate a ese PID.

Ejecución a Listo: Lo que pasa es que se produce una interrupción de timer, la cual cuando concluye el tiempo ejecuta una interupt gate y se deja de ejecutar el proceso en ejecución y es momento dejar que otro proceso tenga una parte del tiempo de la CPU. Se atiende a la rutina de interrupciones del sistema operativo la cual cambia de estado y después llama al planificador para luego poder pasar de listo a ejecución y continuar donde se había quedado el proceso.

Ejecución a Bloqueado y Bloqueado a Listo: Si hay un proceso ejecutándose (usando CPU) y hace entrada/salida se queda bloqueado hasta nuevo aviso, cuando llega de nuevo esa entrada/salida pasa a listo y se queda esperando para ser elegido en algún momento por el planificador. Otra alternativa es que hay un proceso ejecutándose, en donde se le asigna un tiempo determinado y cuando se le acaba el tiempo, llega esa interrupción y lo manda de nuevo a listo.

Ejecución a terminado: Es cuando ejecuta exit o return (0), termina de ejecutarse.



SEÑALES

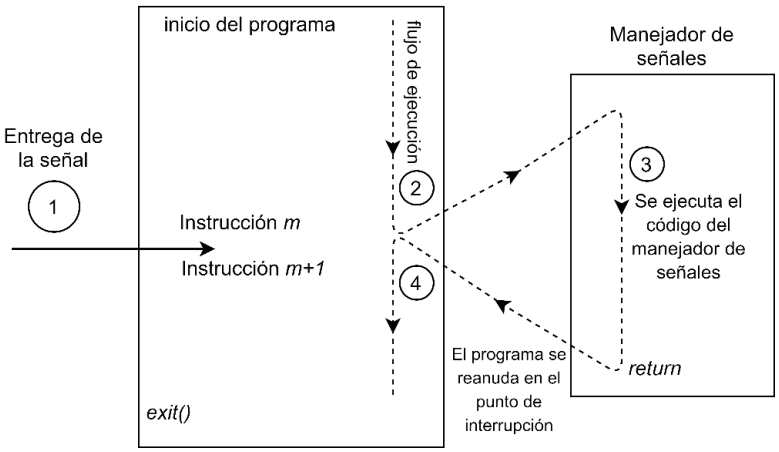
Una señal es una notificación a un proceso de que ha ocurrido un evento. Las señales se describen a veces como interrupciones de software.

Se usan para avisare al sistema operativo de algún evento de software (ej un temporizador se disparó), avisarle al kernel de una excepción de hardware (ej la división por cero), para controlarlo desde el teclado (ej terminar el proceso).

Al recibir una señal, un proceso realiza una de las siguientes acciones por defecto, dependiendo de la señal:

- La señal es ignorada y no tiene ningún efecto sobre el proceso.
- Dejar que se ejecute una acción por defecto relacionada con la señal.
- Ejecutar un manejador para la señal: programa encargado de “hacer algo” con la señal recibida.

Manejador de la señal



Supongamos que en el instante de tiempo en (2) me llega una señal, me va a ejecutar la rutina del manejador (3) y una vez que retorna sigue mi programa (4), parecido a una interrupción solamente que no es de hardware sino de software y es asíncrona.