Procesos – Definición y Estados

OCSO

Profesorado de Informática
CeRP del Suroeste, Uruguay

Contenidos

- Qué es un proceso
- Estructuras de datos para gestionar procesos
- API para trabajar con procesos
- Hilos (threads). Descripción y gestión
- Algoritmos de planificación de la CPU

Introducción y Definiciones Sobre Procesos

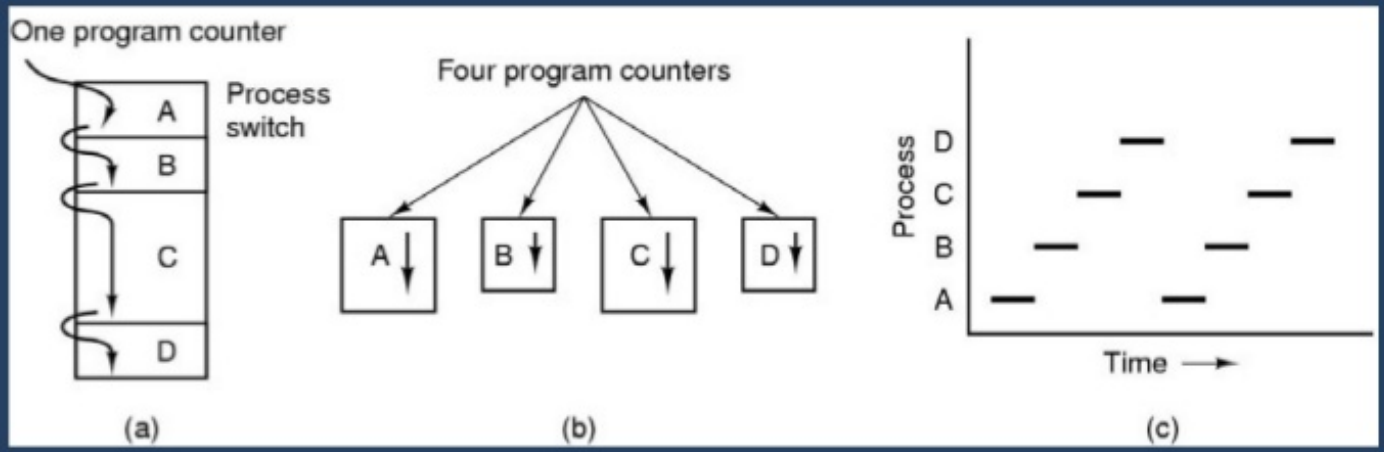
El concepto central de cualquier Sistema Operativo es el de proceso: una abstracción de un programa en ejecución también llamada tarea.

No hay un acuerdo universal sobre una definición de proceso, pero sí algunas definiciones aceptadas.

Un programa que se está ejecutando.
Una actividad asincrónica.
El emplazamiento del control de un procedimiento que está siendo ejecutado.
Aquello que se manifiesta por la existencia en el Sistema Operativo de un bloque de control de proceso.
Aquella entidad a la cual son asignados los procesadores.
La unidad despachable.

En sistemas de multiprogramación la cpu alterna de programa en programa, en un esquema de seudo paralelismo, es decir que la cpu ejecuta en cierto instante un solo programa, intercambiando muy rápidamente entre uno y otro.

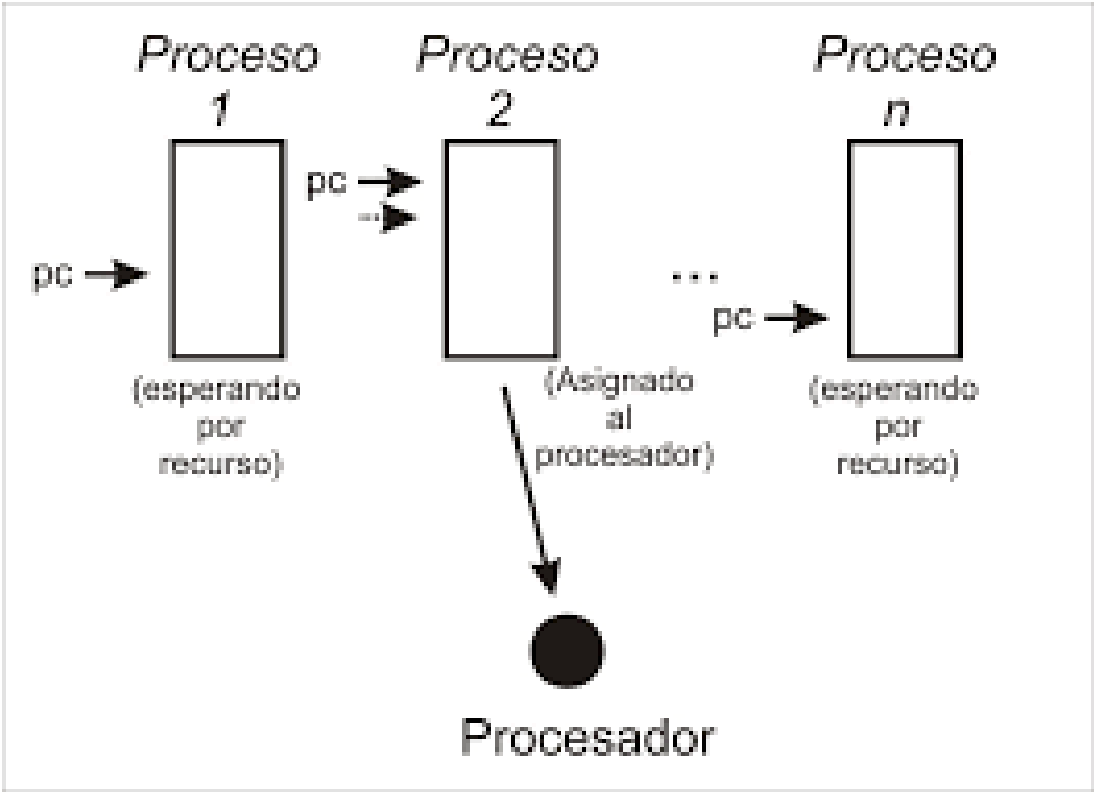
Proceso



(a) Multiprogramación de cuatro programas. (b) Modelo conceptual de cuatro procesos secuenciales independientes. (c) Sólo un programa está activo en cada momento.

Contador de programa

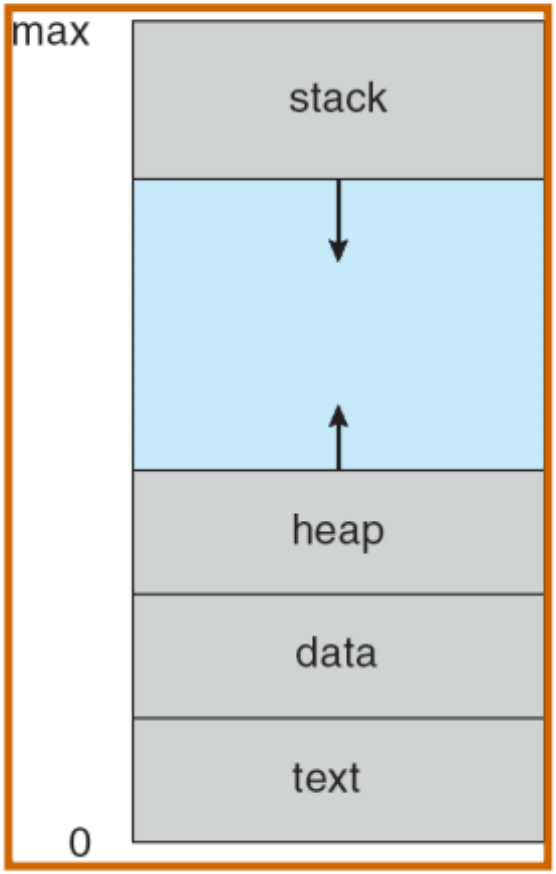
Cada proceso tiene su program counter, y avanza cuando el proceso tiene asignado el recurso procesador. A su vez, a cada proceso se le asigna un número que lo identifica entre los demás: identificador de proceso (process id)



Memoria de los procesos

Un proceso en memoria se constituye de varias secciones:

– Código (text): Instrucciones del proceso.
– Datos (data): Variables globales del proceso.
– Memoria dinánica (heap): Memoria dinámica que genera el proceso.
– Pila (stack): Utilizado para preservar el estado en la invocación anidada de procedimientos y funciones.



Estado de los Procesos

Respecto de los estados del proceso deben efectuarse las siguientes consideraciones:

Cada proceso es una entidad independiente pero frecuentemente debe interactuar con otros procesos.

Los procesos pueden bloquearse en su ejecución porque:

- Desde el punto de vista lógico no puede continuar porque espera datos que aún no están disponibles.
- El Sistema Operativo asignó la cpu a otro proceso.

Los estados que puede tener un proceso son

- En ejecución: utiliza la cpu en el instante dado.
- Listo: ejecutable, se detiene en forma temporal para que se ejecute otro proceso.
- Bloqueado: no se puede ejecutar debido a la ocurrencia de algún evento externo.
- Son posibles cuatro transiciones entre estos estados.

Estado de los Procesos

Respecto de los estados del proceso deben efectuarse las siguientes consideraciones:

Cada proceso es una entidad independiente pero frecuentemente debe interactuar con otros procesos.

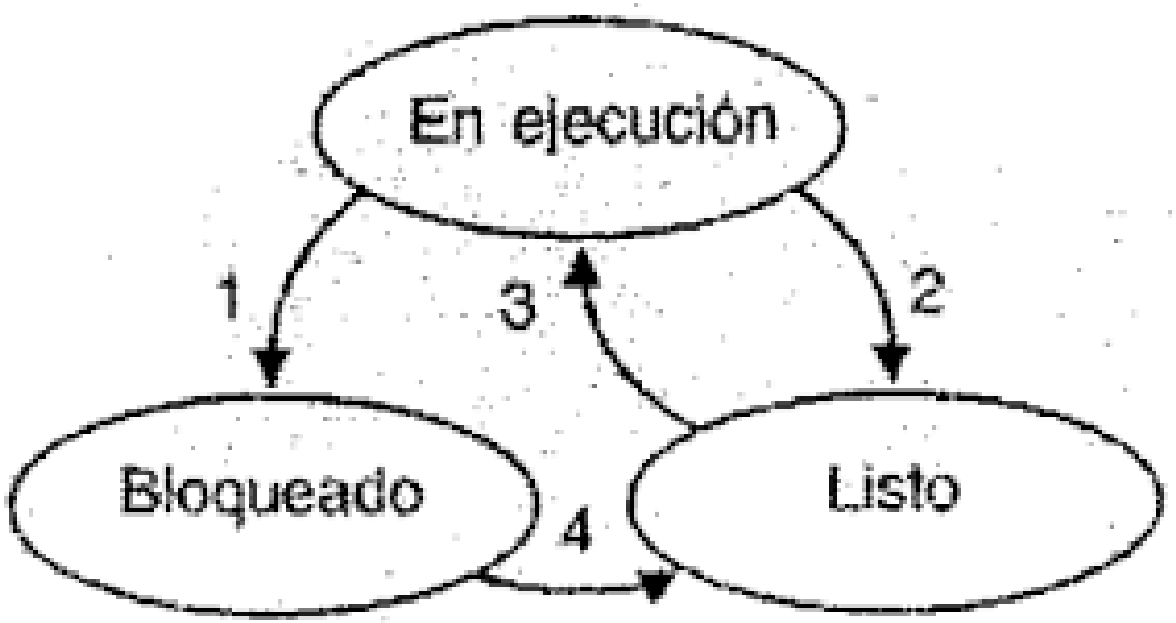
Los procesos pueden bloquearse en su ejecución porque:

- Desde el punto de vista lógico no puede continuar porque espera datos que aún no están disponibles.
- El Sistema Operativo asignó la cpu a otro proceso.

Los estados que puede tener un proceso son

- En ejecución: utiliza la cpu en el instante dado.
- Listo: ejecutable, se detiene en forma temporal para que se ejecute otro proceso.
- Bloqueado: no se puede ejecutar debido a la ocurrencia de algún evento externo.
- Son posibles cuatro transiciones entre estos estados.

Estado de los Procesos



Estado de los Procesos

Durante su existencia un proceso pasa por una serie de estados discretos, siendo varias las circunstancias que pueden hacer que el mismo cambie de estado.

Debido a ello se puede establecer una “Lista de Listos” para los procesos “listos” y una “Lista de Bloqueados” para los “bloqueados”.

La “Lista de Listos” se mantiene en orden prioritario y la “Lista de Bloqueados” está desordenada, ya que los procesos se desbloquean en el orden en que tienen lugar los eventos que están esperando. Al admitirse un trabajo en el sistema se crea un proceso equivalente y es insertado en la última parte de la “Lista de Listos”.

La asignación de la cpu al primer proceso de la “Lista de Listos” se denomina “Despacho”, que es ejecutado por una entidad del Sistema Operativo llamada “Despachador”.

El “Bloqueo” es la única transición de estado iniciada por el propio proceso del usuario, puesto que las otras transiciones son iniciadas por entidades ajenas al proceso.

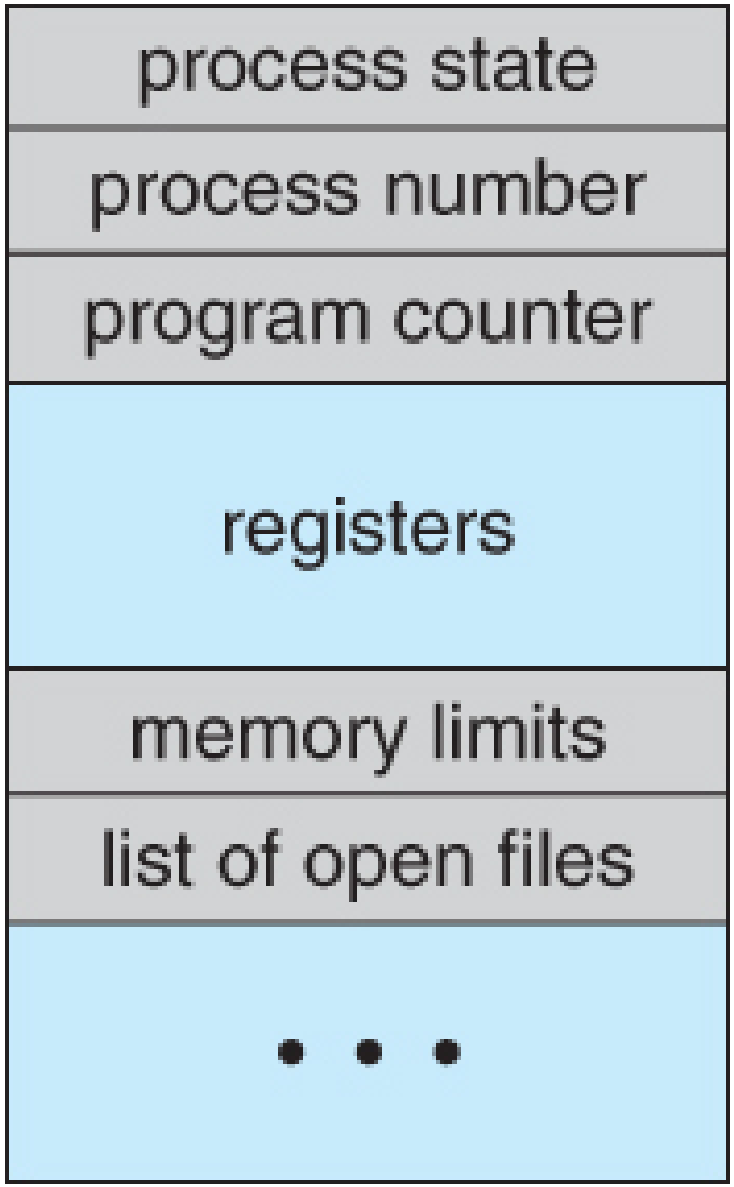
Bloque de Control de Proceso (PCB)

La manifestación de un proceso en un Sistema Operativo es un "Bloque de Control de Proceso" (PCB) con información que incluye:

- Estado actual del proceso.
- Identificación única del proceso.
- Prioridad del proceso.
- Apuntadores para localizar la memoria del proceso.
- Apuntadores para asignar recursos.
- Área para preservar registros.

Cuando el Sistema Operativo cambia la atención de la cpu entre los procesos, utiliza las áreas de preservación del PCB para mantener la información que necesita para reiniciar el proceso cuando consiga de nuevo la cpu.

Bloque de Control de Proceso (PCB)



Estados de Procesos

Los sistemas que administran los procesos deben poder crear, destruir, suspender, reanudar, cambiar la prioridad, bloquear, despertar y despachar un proceso.

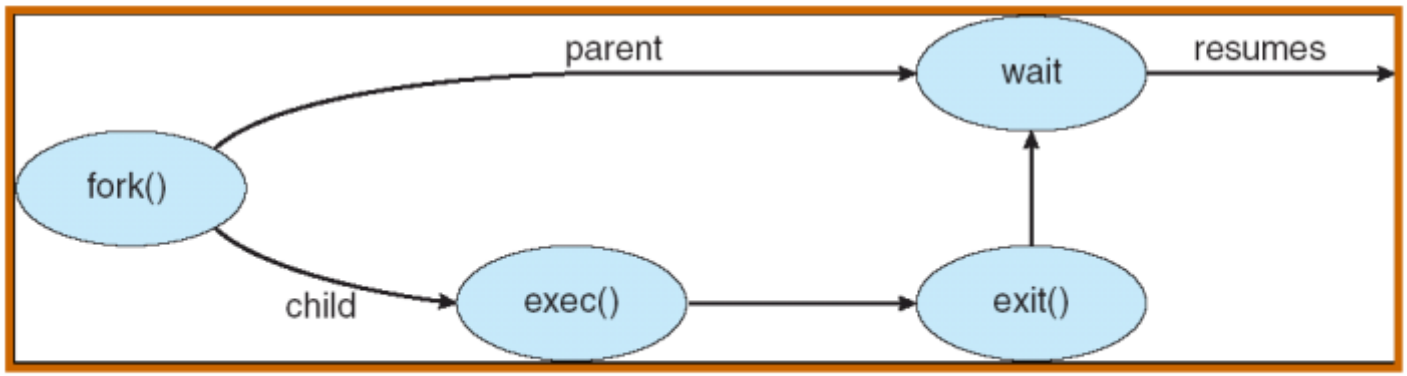
La “creación” de un proceso significa:

Dar nombre al proceso.
Insertar un proceso en la lista del sistema de procesos conocidos.
Determinar la prioridad inicial del proceso.
Crear el bloque de control del proceso.
Asignar los recursos iniciales del proceso.

Un proceso puede “crear un nuevo proceso”, en cuyo caso el proceso creador se denomina “proceso padre” y el proceso creado “proceso hijo” y se obtiene una “estructura jerárquica de procesos”.

Creación de Procesos

Ej.: UNIX
– UNIX provee el system call fork para la creación de un nuevo proceso.
– La invocación a esta función le retorna al padre el número de process id del hijo recién creado y al hijo el valor 0. El hijo comienza su ejecución en el retorno del fork.
– Además, se provee del system call exec que reemplaza el espacio de memoria del proceso por uno nuevo



Creación de procesos y herencia

El hijo HEREDA algunos aspectos del padre y otros no.

HEREDA (recibe una copia privada de....)

- El espacio de direcciones lógico (código, datos, pila, etc).
  - La memoria física es nueva, y contiene una copia de la del padre
- La tabla de programación de signals
- Los dispositivos virtuales
- El usuario /grupo (credenciales)
- Variables de entorno

NO HEREDA (sino que se inicializa con los valores correspondientes)

- PID, PPID (PID de su padre)
- Contadores internos de utilización (Accounting)
- Alarmas y signals pendientes (son propias del proceso)

Terminar ejecución/Esperar que termine

La “destrucción” de un proceso implica:

Borrarlo del sistema.
Devolver sus recursos al sistema.
Purgarlo de todas las listas o tablas del sistema.
Borrar su bloque de control de procesos.

Un proceso “suspendido” no puede proseguir hasta que otro proceso lo reanude.

Reanudar (reactivar) un proceso implica reiniciarlo en el punto donde fue suspendido.

La “destrucción” de un proceso puede o no significar la destrucción de los procesos hijos, según el Sistema Operativo.

Terminar ejecución/Esperar que termine

Un proceso puede acabar su ejecución voluntaria (exit) o involuntariamente (signals)

Cuando un proceso quiere finalizar su ejecución (voluntariamente), liberar sus recursos y liberar las estructuras de kernel reservadas para él, se ejecuta la llamada a sistema exit.

Si queremos sincronizar el padre con la finalización del hijo, podemos usar waitpid:
El proceso espera (si es necesario se bloquea el proceso) a que termine un hijo cualquiera o uno concreto

- waitpid(-1,NULL,0) → Esperar (con bloqueo si es necesario) a un hijo cualquiera
- waitpid(pid_hijo,NULL,0)→ Esperar (con bloqueo si es necesario) a un hijo con pid=pid_hijo

El hijo puede enviar información de finalización (exit code) al padre mediante la llamada a sistema exit y el padre la recoge mediante wait o waitpid

- El SO hace de intermediario, la almacena hasta que el padre la consulta
- Mientras el padre no la consulta, el PCB no se libera y el proceso se queda en estado ZOMBIE (defunct)

Conviene hacer wait/waitpid de los procesos que creamos para liberar los recursos ocupados del kernel
Si un proceso muere sin liberar los PCB's de sus hijos el proceso init del sistema los libera

Ejemplos en c++

Crear, eliminar y entender de forma práctica los procesos zombie / fantasma en Unix

https://goo.gl/c9D0PS

Manual básico de administración de procesos

http://goo.gl/ZJlmAk

Prácticas de procesos

http://goo.gl/zDdvEB