Procesos

Procesos

- Concepto de Proceso
- Planificación de Proceso
- Operaciones sobre Procesos
- Comunicaciones Interprocesos (IPC)
- Ejemplos de Sistemas de IPC
- Comunicación en un Sistema Cliente-Servidor

Concepto de Proceso

- Un SO ejecuta una variedad de programas:
  - Sistema Batch – jobs
  - Sistemas de Tiempo Compartido – programas de usuario o tareas
- **Proceso** – un programa en ejecución.

- Un proceso incluye:
  - contador de programa
  - stack
  - sección de datos

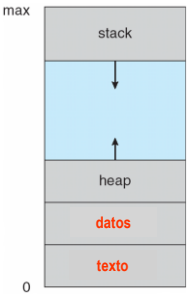


Diagrama de Estados de un Proceso – 3 estados

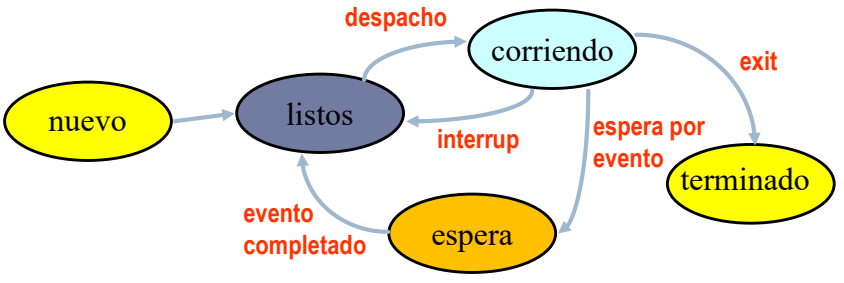
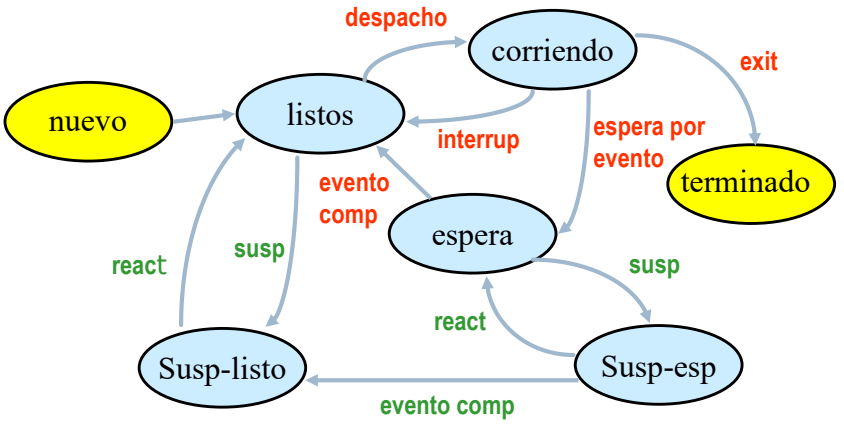


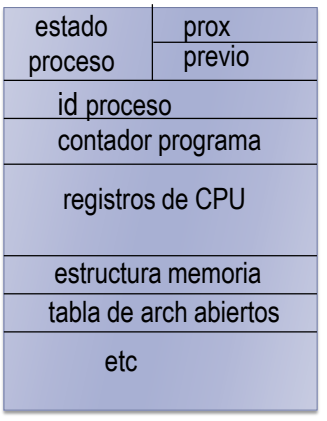
Diagrama de Estados de un Proceso – 5 estados



Bloque de Control de Procesos (PCB)

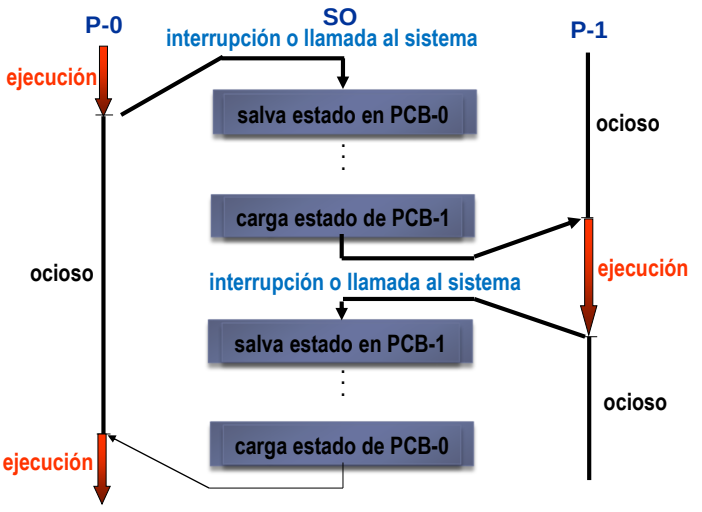
Es una estructura de dato que contiene información asociada con cada proceso.

- Estado de Proceso
- Contador de Programa
- Registros de CPU
- Información de planificación de CPU
- Información de administración de memoria
- Información contable
- Información de estado E/S



PCB: Process Control Block

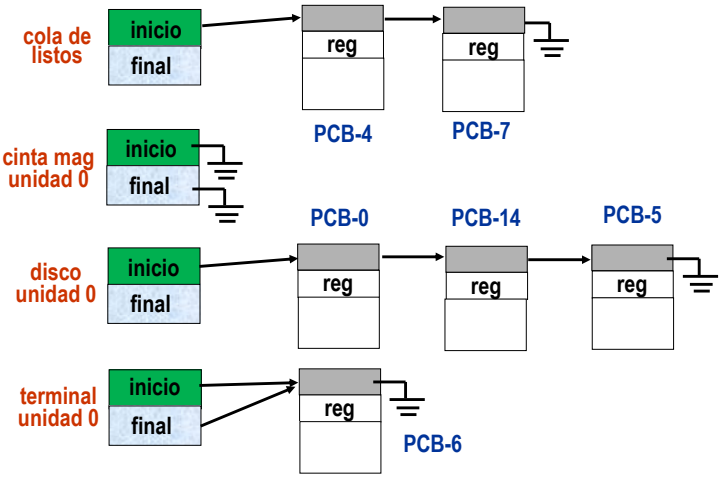
Conmutación de CPU de Proceso a Proceso



Colas de Planificación de Procesos

- Cola de Job (o tareas) – conjunto de todos los procesos en el sistema.
- Cola de listos – conjunto de todos los procesos residentes en memoria principal, listos y esperando para ejecutar.
- Colas de dispositivos – conjunto de procesos esperando por una E/S en un dispositivo de E/S.
- Migración de procesos entre las colas.

Colas de listos y de Dispositivos de E/S



Planificadores de Procesos

- Planificador de **largo término** (o planificador de jobs) – selecciona que procesos deberían ser puestos en la cola de listos.
- Planificador de **corto término** (o planificador de CPU) – selecciona que procesos deberían ser proximamente ejecutados y colocados en la CPU.
- Planificador de **mediano término**

Planificadores de Procesos

- El planificador de corto término es invocado muy frecuentemente (milisegundos) ⇒ (debe ser rápido).
- El planificador de largo término es invocado poco frecuentemente (segundos, minutos) ⇒ (puede ser muy lento).
- El planificador de largo término controla el *grado de multiprogramación*.
- Los procesos pueden ser descriptos como:
  - ***Procesos limitados por E/S***
  - ***Procesos limitados por CPU***

Cambio de contexto

- Cuando la CPU conmuta a otro proceso, el sistema debe salvar el estado del viejo proceso y cargar el estado para el nuevo proceso vía un cambio de contexto.
- El contexto de un proceso está representado en el PCB
- El tiempo que lleva el cambio de contexto es sobrecarga; el sistema no hace trabajo útil mientras está conmutando.
- El tiempo depende del soporte de hardware.

Creación de Procesos

Actividades

1. Asignar un identificar de proceso único al proceso.
2. Reservar espacio para proceso.
3. Inicialización del PCB.
4. Establecer los enlaces apropiados.
5. Creación o expansión de otras estructuras de datos.

Creación de Procesos – Políticas

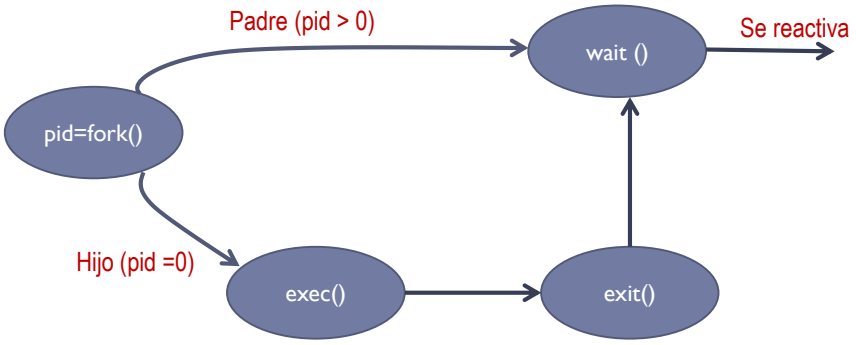
- Espacio de direcciones
  - El hijo duplica el del padre.
  - El hijo tiene un programa cargado en él.
- Recursos compartidos
  - Padres e hijos comparten todos los recursos.
  - Hijo comparte un subconjunto de los recursos del padre.
  - Padre e hijo no comparten ningún recurso.
- Ejecución
  - Padres e hijos ejecutan concurrentemente.
  - Padres esperan hasta que los hijos terminan.

Creación de Procesos

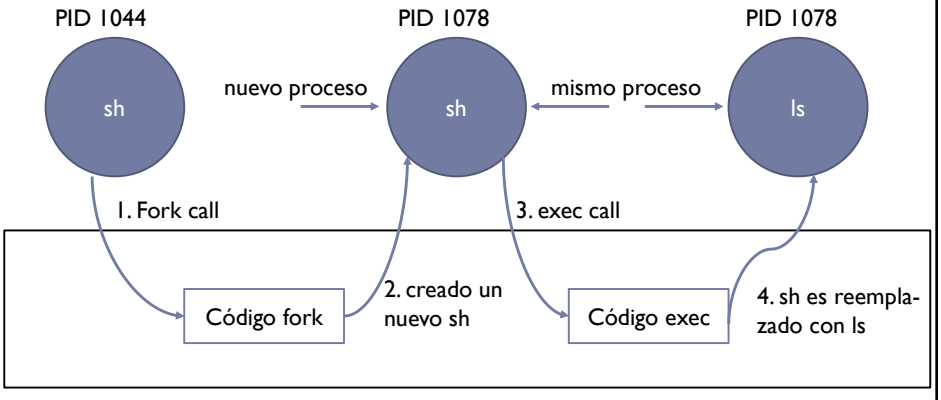
Ejemplos de UNIX

La llamada a sistema **fork** crea un nuevo proceso

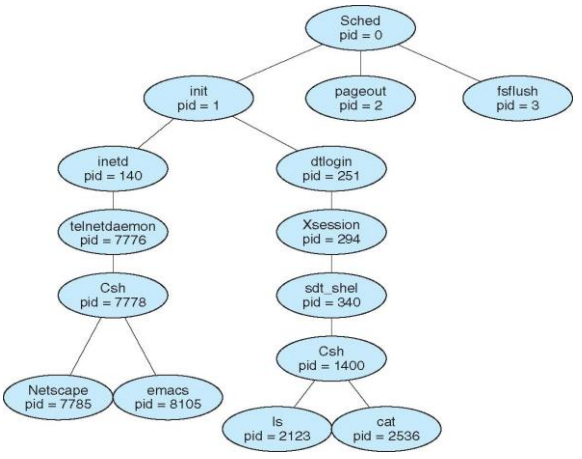
La llamada a sistema **exec** es usada después del **fork** para reemplazar el espacio de memoria del proceso con un nuevo programa.



Creación de Procesos en UNIX - Ejemplo



Árbol de Procesos en UNIX



Terminación de Procesos

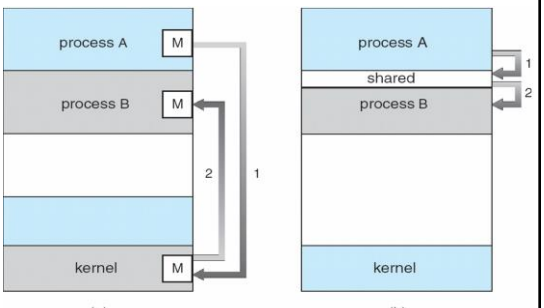
- El proceso ejecuta la última sentencia y espera que el SO haga algo (**exit**).
  - Los datos de salida del hijo se pasan al padre (vía **wait**).
  - Los recursos de los procesos son liberados por el SO.
- El padre puede terminar la ejecución del proceso hijo (**abort**).
  - El hijo ha excedido los recursos alocados.
  - La tarea asignada al hijo no es mas requerida.
  - El padre está terminando.
    - El SO no permite a los hijos continuar si su padre termina.
    - Terminación en cascada.

Procesos Cooperativos

- Un proceso ***independiente*** no puede afectar ni ser afectado por la ejecución de otro proceso.
- Un proceso ***cooperativo*** puede afectar o ser afectado por la ejecución de otro proceso.
- Ventajas de los procesos cooperativos
  - Información compartida
  - Aceleración de la computación
  - Modularidad
  - Conveniencia

Comunicación Interprocesos

- Los procesos cooperativos necesitan **comunicación interprocesos** (**IPC**)
- Dos modelos de IPC
  - Memoria compartida
  - Pasaje de Mensajes



(a) Pasaje de Mensaje

(b) Memoria Compartida

Problema del Productor-Consumidor

- Paradigma procesos cooperativos, el proceso ***productor*** produce información que es consumida por un proceso ***consumidor*** .
  - *buffer ilimitado* - no tiene límites prácticos en el tamaño del buffer.
  - *buffer limitado* supone que hay un tamaño fijo de buffer.

Modelos de Comunicación – Memoria Compartida

Ejemplo de procesos cooperativos: Productor-Consumidor

```
while (true) {
     /* produce an item in next_produced */

     while (((in + 1) % BUFFER_SIZE) == out)
        ; /* do nothing */

     buffer[in] = next_produced;
     in = (in + 1) % BUFFER_SIZE;
}
```

```
#define BUFFER_SIZE 10

typedef struct {
    . . .
}item;

item buffer[BUFFER_SIZE];
int in = 0;
int out = 0;
```

```
while (true) {
     while (in == out)
        ; /* do nothing */

     next_consumed = buffer[out];
     out = (out + 1) % BUFFER_SIZE;

     /* consume the item in next_consumed */
}
```

Comunicación entre Procesos (IPC)

- Sistema de mensajes – los procesos se comunican uno con otro sin necesidad de variables compartidas.
- Provee dos operaciones:
  - **send**(*mensaje*)
  - **receive**(*mensaje*)
- Si *P* and *Q* desean comunicarse, necesitan:
  - Establecer un *vínculo de comunicación* entre ellos
  - Intercambiar mensajes via send/receive
- Implementación de un vínculo de comunicación
  - lógico (p.e., propiedades lógicas)
  - físico (p.e., memoria compartida, canal hardware)

Comunicación Directa

- Los procesos deben nombrar al otro explícitamente:
  - **send** (*P, mensaje*) – envía un mensaje al proceso P
  - **receive**(*Q, mensaje*) – recibe un mensaje del proceso Q
- Propiedades del vínculo de comunicación
  - Un vínculo está asociado con exactamente un par de procesos que se comunican.
  - Entre cada par de procesos existe exactamente un vínculo.
  - El vínculo puede ser unidireccional, pero es usualmente bi-direccional.

Comunicación Indirecta

- Los mensajes son dirigidos y recibidos desde ***mailboxes***
- Vínculo de comunicación
  - Se establece solo si los procesos comparten un mailbox común.
  - Puede ser asociado con muchos procesos.
  - Cada par de procesos puede compartir varios vínculos de comunicación.
  - Puede ser unidireccional o bi-direccional.
- Operaciones
  - crear un nuevo *mailbox*
  - enviar y recibir mensajes por medio del *mailbox*
  - destruir un *mailbox*
- Las primitivas son:
  send(*A, message*) – enviar un mensaje al mailbox A
  receive(*A, message*) – recibir un mensaje del mailbox A

Sincronización

- El pasaje de mensajes puede ser bloqueante o no bloqueante.

- **Bloqueante** es considerado **sincrónico**
  - ***Send bloqueante***
  - ***Receive bloqueante***
- **No bloqueante** es considerado **asincrónico**
  - ***Send no bloqueante***
  - ***Receive no bloqueante***

Buffering

- La cola de mensajes asociada al vínculo se puede implementar de tres maneras.
  1. Capacidad – 0 mensajes
     El enviador debe esperar por el receptor (*rendez-vous*).
  2. Capacidad limitada – longitud finita de *n* mensajes
     El enviador debe esperar si el vínculo está lleno.
  3. Capacidad ilimitada – longitud infinita
     El enviador nunca espera.

Comunicación Cliente-Servidor

- Sockets
- Llamadas a Procedimientos Remotos (**RPC**:Remote Procedure Calls)
- Invocación a Métodos Remotos (**RMI**:Remote Method Invocation (Java))

Bibliografía:

- Silberschatz, A., Gagne G., y Galvin, P.B.; "*Operating System Concepts*", 7ma Edición 2009; 9na Edición 2012; 10ma Edición 2018.
- Stallings, W. *"Operating Systems: Internals and Design Principles"*, Prentice Hall, 5ta Edición 2005; 6ta Edición 2009; 7maEdición 2011; 9na Edición 2018.
- Tanenbaum, A.; "*Modern Operating Systems*", Addison-Wesley, 3ra. Edición 2008, 4ta. Edición 2014.