Sistemas Operativos

Tema 2 Administración de Procesos

Ing. Rafael Sandoval Vázquez
sistem.operativos@gmail.com

2. Administración de Procesos

Objetivo:

El alumno identificará los tipos de procesos y sus estados. Describirá las diferentes técnicas de comunicación y sincronización de procesos concurrentes.

Procesos

Todos los programas, cuya ejecución solicitan los usuarios, se ejecutan en forma de procesos. **El proceso se puede definir como un programa en ejecución y, de una forma un poco más precisa como la unidad de procesamiento gestionada por el sistema operativo.**

Procesos

Un proceso es:

- Sección de texto (código del programa)
- Actividad actual, representada por:
  - Registros de memoria
  - Estado

Procesos

- Incluye además
  - - Pila (stack), que contiene datos temporales (parámetros de subrutinas, direcciones de retorno y variables locales).
  - - Sección de datos, que contiene variables globales y memoria dinámica.

Procesos

Para que un programa pueda ser ejecutado ha de residir con sus datos en memoria principal.

Procesos

Un proceso necesita ciertos recursos para realizar satisfactoriamente su tarea:

- Tiempo de CPU
- Memoria
- Archivos
- Dispositivos de E/S

Procesos

- Estos recursos se asignan a un proceso:
  - Cuando se crea
  - Durante su ejecución.

Procesos

- Las obligaciones del sistema operativo como gestor de procesos son:
  - -Creación y eliminación de procesos
  - -Planificación de procesos (procurando la ejecución de múltiples procesos maximizando la utilización del procesador)
  - -Establecimiento de mecanismos para sincronización y comunicación de procesos.
  - -Manejo de bloqueos mutuos.

Procesos

- El sistema operativo mantiene por cada proceso una serie de estructuras de información que permiten identificar las características de éste, así como los recursos que tiene asignados, en esta última categoría entran los descriptores de los segmentos de memoria asignados, los descriptores de los archivos abiertos, los descriptores de los puertos de comunicación, etc.

Procesos

- Una parte muy importante de esta información se encuentra en el llamado bloque de control del proceso (BCP). El sistema operativo mantiene una tabla de procesos con todos los BCP de los procesos.

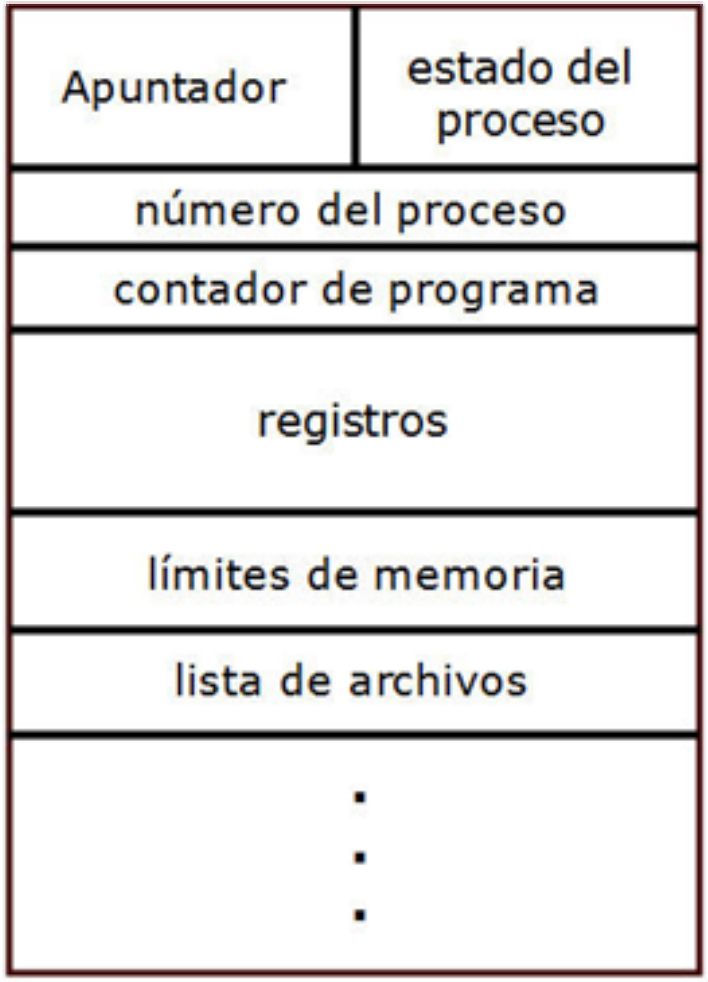
Procesos

- Por razones de eficiencia la tabla de procesos se construye normalmente como una estructura estática, que tiene un determinado número de BCP, todos ellos del mismo tamaño. De manera general podemos decir que la información que compone un proceso es:

Procesos

- - Contenido de los segmentos de memoria en los que residen el código y los datos del proceso. (imagen de memoria ó *core image*)
- - Contenido de los registros del modelo de programación.
- - Contenido del BCP.

Bloque de Control de Proceso



BCP

- Cada proceso se presenta en el SO con un bloque de control de proceso (PCB, *Process Control Block*), también llamado bloque de control de tarea.
- Este bloque contiene muchos elementos de información asociados a un proceso específico, incluidos los siguientes:

BCP

- - Estado del proceso

- - Contador de Programa.- El contador indica la dirección de la siguiente instrucción que se ejecutará para este proceso.

BCP

- - Registros de CPU.- Varían dependiendo de la computadora, incluyen acumuladores, registros índice, apuntadores de pila y registros de propósito general.
- - Información de planificación de CPU.- Incluye una prioridad del proceso, apuntadores a colas de planificación y cualquier otro parámetro de planificación que haya.

BCP

- -Información de administración de la memoria.- Puede incluir datos tales como el valor de los registros de base y límite, las tablas de páginas o las tablas de segmentos, dependiendo del sistema de memoria empleado por el SO.

BCP

- - Información contable.- Incluye la cantidad de tiempo de CPU y tiempo real consumida, límites de tiempo, números de cuenta, números de trabajo o proceso, y demás.
- - Información de E/S. La información incluye una lista de dispositivos de E/S asignadas a este proceso, una lista de archivos abiertos, etc.

Estados básicos de un proceso

- Cuando se admite una tarea en el sistema, se crea el proceso correspondiente y se inserta normalmente al final de la lista de procesos listos.
- El proceso se desplaza poco a poco hacia el frente de la lista de procesos listos.

Estados básicos de un proceso

- Cuando el proceso llega al principio de la lista, se le asigna la CPU cuando esta queda disponible y entonces se dice que hay una “transición de estado”, del estado listo al estado de la ejecución.

Estados básicos de un proceso

- La asignación del procesador al primer proceso de la lista de procesos listos se denomina despacho; dicha actividad la realiza una entidad del sistema llamada despachador, (dispatcher).

QUANTUM

- Para evitar que un proceso monopolice el sistema (en forma accidental o malintencionada), el SO utiliza un reloj de interrupción por hardware (cronómetro de intervalos) para que las tareas de un usuario se ejecuten durante un intervalo específico de tiempo (quantum).

Estados básicos de un proceso

- - Si el proceso no libera voluntariamente la CPU antes de que expire el intervalo de tiempo, el reloj genera una interrupción, haciendo que retome el control el SO.
- - El SO transforma entonces el proceso que estaba ejecutándose en un proceso listo y procede a ejecutar el primero de los procesos de la lista de listos.

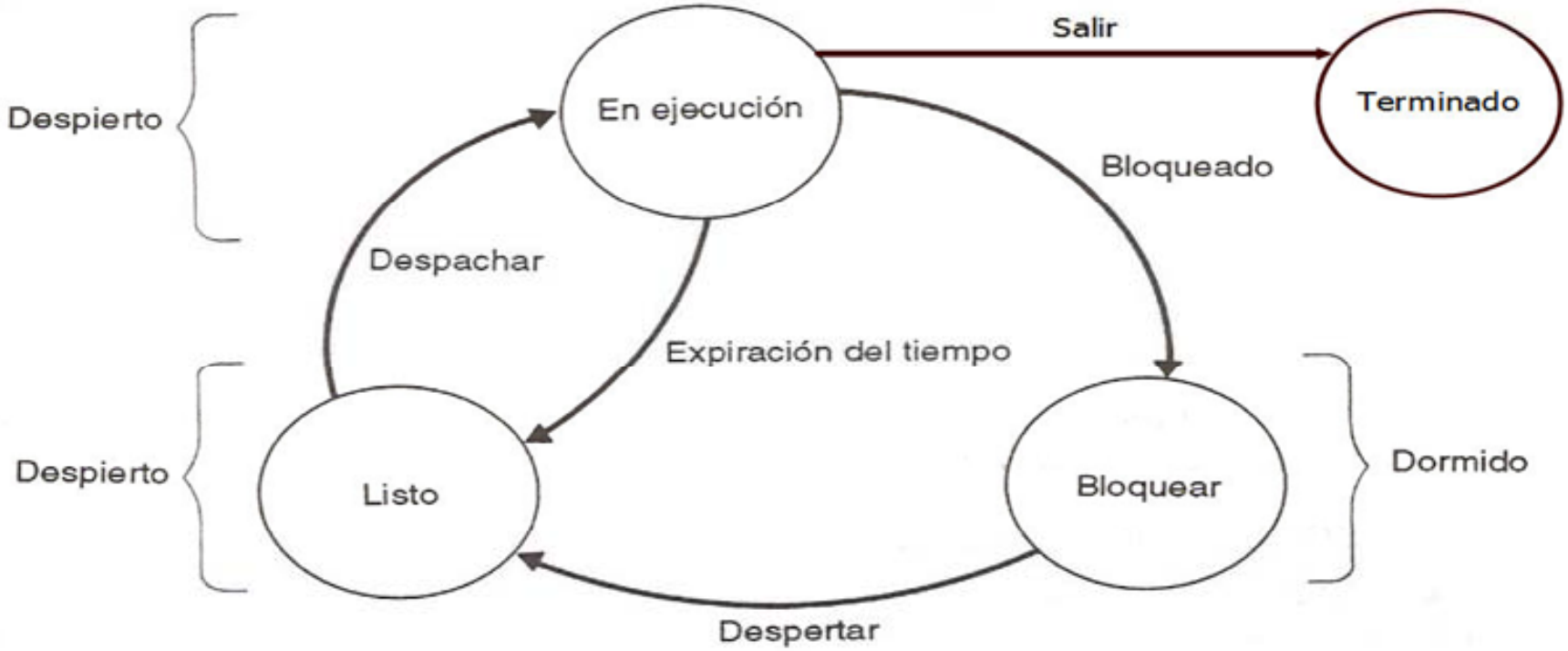
Estados básicos de un proceso

- Si el proceso que se está ejecutando inicia una operación de E/S antes de que expire su cuanto (quantum), libera voluntariamente la CPU (es decir el proceso se bloquea a sí mismo esperando a que se complete una operación de E/S).

Estados básicos de un proceso

La última transición posible es este modelo de tres estados ocurre cuando se completa una operación de E/S (o algún otro evento que espere un proceso). El proceso realiza la transición del estado bloqueado al estado listo.

Estados básicos de un proceso



Estados básicos de un proceso

Se han definido entonces cuatro posibles transiciones de estado:

- Despachar(proceso):**listo->en_ejecución**
- Tiempo_expirado(proceso):**en_ejecución->listo**
- Bloquear(proceso):**en_ejecución->bloqueado**
- Despertar(proceso):**bloqueado->listo**

Estados básicos de un proceso

PID	ESTADO
01	L
02	L
03	B
07	E
04	L
05	B
09	B

Estados básicos de un proceso

Ejercicio:

Indique los estados y la transición que presenta el proceso si:

- Tiempo expirado(07):
- Termina tarea de E/S(05):
- Llega a la parte superior de la cola de listos(04):
- Operación completada de E/S(09):
- Solicitud de Lectura de floppy(04):
- Inicia lectura de cdrom(03):
- Termina lectura de floppy(04):

Creación de un proceso

- **Creación de un proceso**
- Para la creación de un proceso tenemos las siguientes actividades:

- Nuevo trabajo por lotes.- El SO se prepara para escoger un nuevo trabajo (cinta o disco), leerá la próxima secuencia de órdenes de control de trabajos.

Creación de un proceso

- - Conexión interactiva.- Un usuario entra en el sistema desde una Terminal.
- - Creado por el SO para dar un servicio.- El SO puede crear un proceso para llevar a cabo una función de parte de un programa de usuario, sin que el usuario tenga que esperar.
- - Generado por un proceso existente.- Permite que un proceso pueda originar la creación de otro.

Terminación de un proceso

La terminación de un proceso se puede dar debido a:

- Terminación normal
- Tiempo límite excedido
- No hay memoria suficiente
- Violación de límites
- Error de protección
- Error aritmético
- Tiempo máximo de espera rebasado
- Error de E/S
- Instrucción ilegal
- Intervención del operador o del SO
- Terminación del padre
- Solicitud del padre

Estados de procesos

- Solamente puede haber un proceso en ejecución a la vez, pero puede haber varios listos y bloqueados. La lista de procesos listos se ordena por prioridad de manera que el siguiente proceso que reciba la CPU será el primero de la lista.

Estados de procesos

- La lista de procesos bloqueados normalmente no está ordenada, los procesos no se desbloquean en orden de prioridad, sino que lo hacen en el orden de ocurrencia de los eventos que están esperando.

Estados de procesos

- Hay situaciones en las cuales varios procesos pueden bloquearse esperando la ocurrencia del mismo evento; en tales casos es común asignar prioridades a los procesos que esperan.

Estados de procesos

Dado lo anterior y algunas otras necesidades es necesario contar con un nuevo estado: **Suspendido**

Las razones por las cuales un proceso se puede suspender son:

- Intercambio: El sistema operativo necesita liberar suficiente memoria principal para cargar un proceso que está listo para ejecutarse.

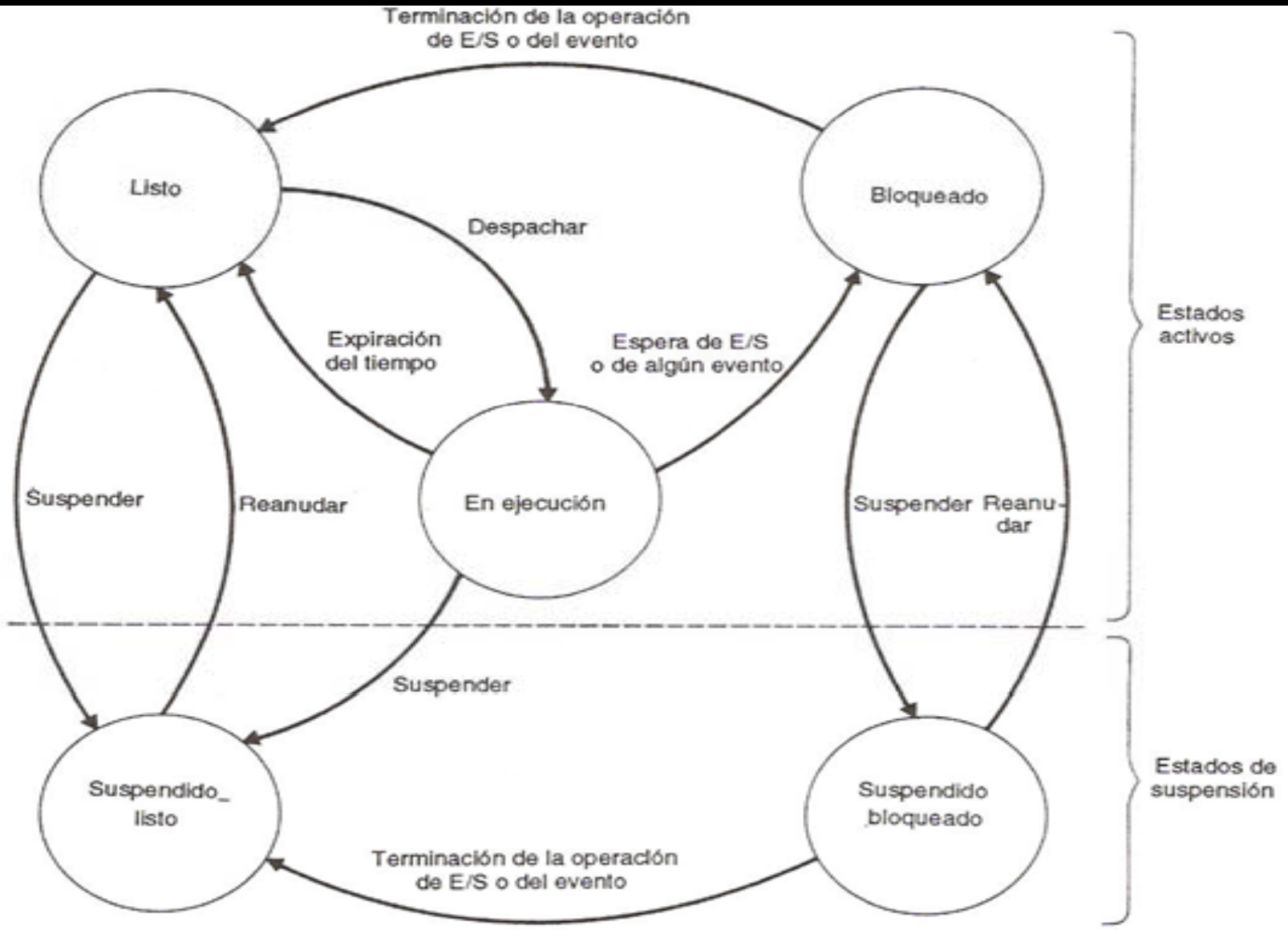
Estados de procesos

- Otra razón del SO: El SO puede suspender un proceso subordinado, ó a un proceso que se sospecha sea causante de un problema.
- Solicitud de un usuario interactivo: Un usuario puede querer suspender la ejecución de un programa con fines de depuración o en conexión con el uso de un recurso.

Estados de procesos

- Temporización: Un proceso puede ejecutarse periódicamente y puede ser suspendido mientras espera el siguiente intervalo de tiempo.
- Solicitud del proceso padre: Un proceso padre puede querer suspender la ejecución de un descendiente para examinar o modificar el proceso suspendido o para coordinar la actividad de varios descendientes.

Estados de procesos



Estados de procesos

Considerando la suspensión, las nuevas transiciones entre estado son:

- **-Suspender : En ejecución → Suspendido_listo**

  Se suspende el proceso actual en ejecución.

- **-Reanudar : Suspendido_listo → Listo**

  El proceso es reanudado por el sistema operativo y pasa a la cola de procesos listos para ejecución.

Estados de procesos

- **-Suspender: Listo → Suspendido_listo**

  Se produce la suspensión del proceso en estado listo por parte del sistema operativo. En un sistema con un solo procesador, el proceso en ejecución puede suspenderse a sí mismo. En un sistema de múltiples procesadores, un proceso en ejecución puede suspender a otro que se esté ejecutando en ese mismo momento en un procesador diferente.

Estados de procesos

- **Suspender : Bloqueado → Suspendido_bloqueado**

  Se suspende el proceso que estaba a la espera de algún evento o una operación de entrada/salida.

Estados de procesos

- **Reanudar : Suspendido_bloqueado → Bloqueado**

  El proceso es reanudado antes de que el evento que espera suceda. Esta transición no se da en algunos sistemas, cuando un proceso bloqueado se suspende no se reanuda hasta que el evento suceda, y entonces pasa a Suspendido_listo.

Estados de procesos

- **Despertar : Suspendido_bloqueado → Supendido_listo**

- El evento que se espera, sucede. El proceso no se reanuda sino que se queda en Suspendido_listo. Cuando se reanude ya no estará bloqueado.

Información en el sistema de un proceso

Un proceso se identifica con un PID (entero), que es la entrada a una tabla del sistema:

Tabla de procesos activos:

- Estado del proceso y datos de su planificación.
- Campo de localización del proceso en memoria principal y secundaria.
- Identificador del usuario al que pertenece y del grupo de usuarios.

Información en el sistema de un proceso

- Identificadores de otros procesos relacionados con él.
- Datos de los eventos que le harán despertarse y señales pendientes.
- Tamaño del proceso, Temporizadores del uso de recursos, etc.
- Puntero al área de usuario.

Información en el sistema de un proceso

- **Área de usuario (cuando se ejecuta el proceso):**

  – Entrada a la tabla de procesos activos.
  – Identificaciones del usuario (real y efectivo).
  – Tiempos de ejecución en modo usuario y kernel.
  – Directorios de trabajo, Descriptores de los archivos abiertos, Terminal original.

Creación de un proceso desde un programa

Para usar el sistema se utilizan las system calls:

- – **fork:** Usada para crear un nuevo proceso mediante la duplicación del proceso llamador. fork es la primitiva básica de creación de procesos.
- – **exec:** Una familia de llamadas al sistema, cada una de las cuales realiza la misma función: la transformación de un proceso mediante el recubrimiento de su espacio de memoria con un nuevo programa.

Creación de un proceso desde un programa

- – **wait:** Permite que un proceso espere a que otro, relacionado con él (por ejemplo un hijo), termine. Esta llamada es un método rudimentario de sincronización entre procesos.
- – **exit:** Se utiliza para terminar un proceso, forzando a cerrar todos los archivos del proceso. Hace re-arrancar a un proceso padre que espera la terminación de un hijo.

Creación de un proceso desde un programa

- Lamina Fork

exit y wait : procesos especiales

En la utilización de *exit* y *wait* se pueden dar dos casos especiales:

- Un hijo llama a *exit* cuando su padre no está en ese momento ejecutando *wait*.

- Un padre llama a *exit*, mientras uno o varios hijos aún están ejecutándose.

exit y wait : procesos especiales

- • En estos dos casos, tendremos dos procesos “especiales”:
- – **zombi:** proceso muerto sin consumir recursos que espera a que su padre haga un *wait*.
- – **huérfano:** proceso con su padre muerto que es adoptado por el
- proceso *init* (1).

Identificaciones

- Los procesos como los usuarios o los archivos se agrupan en grupos de procesos,
- “familia”.
- Se pueden conseguir varias identificaciones con llamadas al sistema:

  – El identificador del proceso PID: **getpid**()
  – El identificador del proceso padre: **getppid**()
  – El identificador del grupo GID: **getgrp**()

Identificaciones

- El grupo de procesos lo utiliza el *shell* para matar a todos los procesos que se han quedado “colgados” o se están ejecutando en esa sesión.
- Los procesos creados desde el *shell* tendrán como padre a el propia *shell*.

EXEC

- Una llamada a exec transforma al proceso “llamador”, cargando un nuevo programa en su espacio de memoria (conserva el mismo PID).

- En POSIX existe una familia de funciones exec

- La familia está formada por: *execl, execle, execve, execlp, execvp y execv.*

EJEMPLO DE EXEC

```
main()
{
        printf("ejecutando ls \n");
        execl("/bin/ls", "ls", "-l", (char *)0);
        perror("execl fallo al intentar ejecutar ls");
        exit(1);
}
```

Tarea DE EXEC

- -Implementar cada uno de los miembros de exec, en programas diferentes.

- -Crear un proceso hijo para lanzar el exec.

Hilos y Multihilos (Threads)

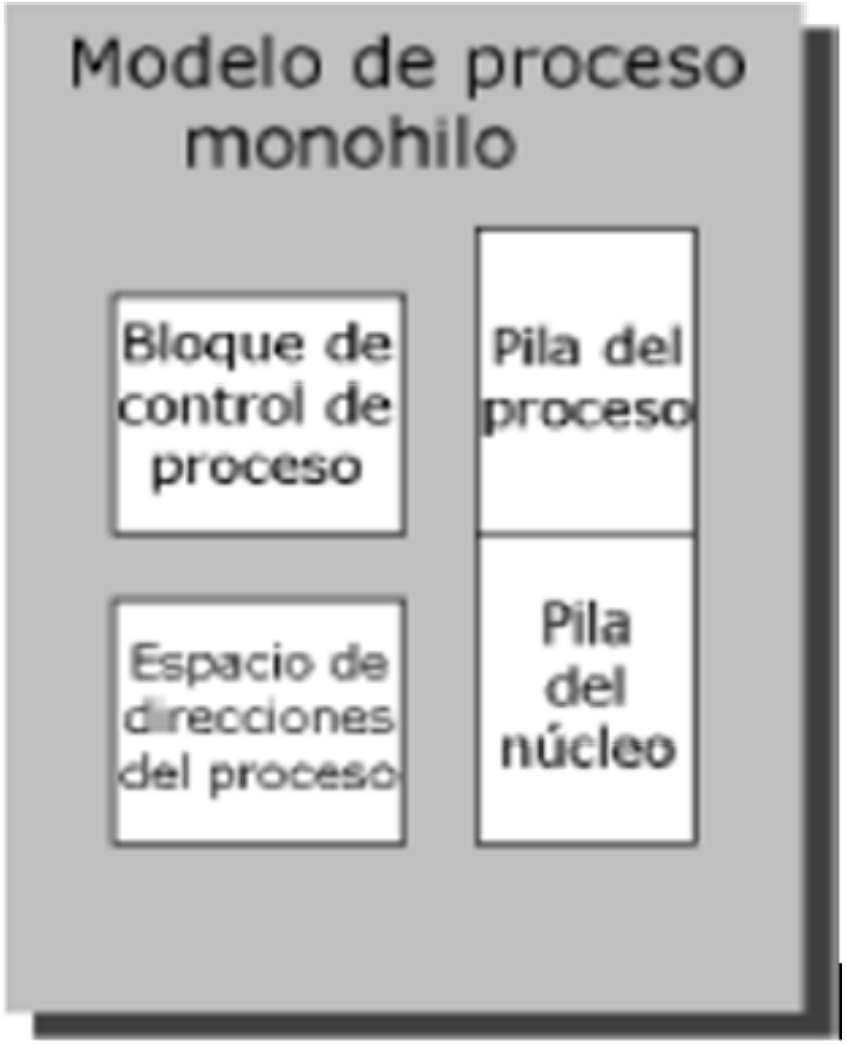
- **Un hilo se puede definir como la traza de ejecución de un proceso.**

- La aparición de los hilos viene justificada desde dos pilares básicos: facilitar la sincronización entre procesos y manejar la eficiencia en la alternancia de procesos en el procesador

Monohilo

- Es un S.O. con procesos monohilo (un solo hilo de ejecución por proceso), no existe el concepto de hilo, la representación de un proceso incluye un espacio de direcciones del proceso, una pila de proceso y una pila núcleo.

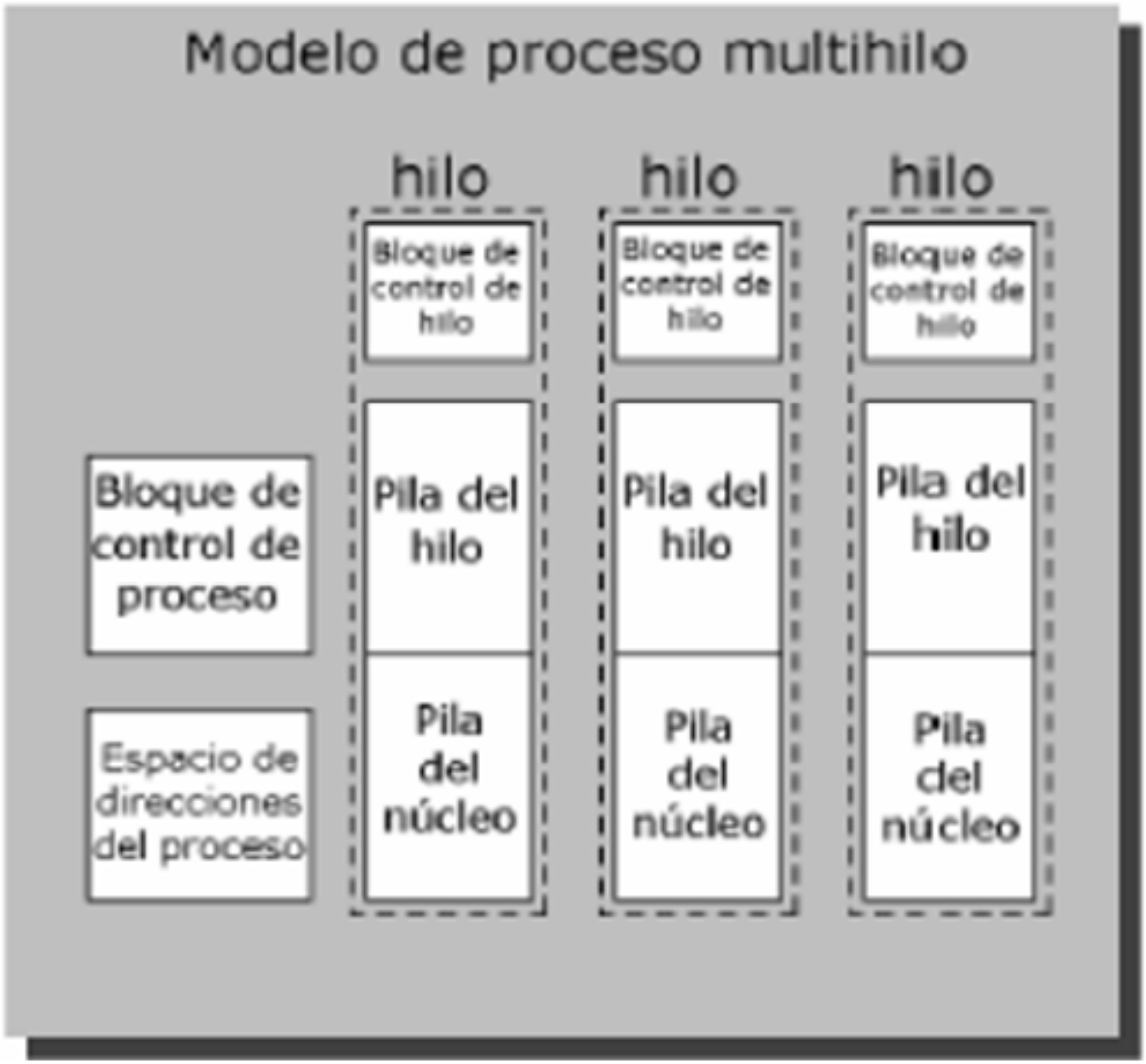
Monohilo


Modelo de proceso monohilo
Bloque de control de proceso
Pila del proceso
Espacio de direcciones del proceso
Pila del núcleo

Multihilo

- -Se pueden programar múltiples hilos de ejecución para que corran simultáneamente en el mismo programa.
- -En un SO con procesos multihilo, solo hay un BCP y un espacio de direcciones asociado al proceso. Sin embargo ahora hay pilas separadas para cada hilo y bloques de control para cada hilo..

Monohilo


Modelo de proceso multihilo
hilo
hilo
hilo
Bloque de control de hilo
Bloque de control de hilo
Bloque de control de hilo
Bloque de control de proceso
Pila del hilo
Pila del hilo
Pila del hilo
Espacio de direcciones del proceso
Pila del núcleo
Pila del núcleo
Pila del núcleo

Casos de S.O. con hilos

- Dentro de la categoría de los sistemas monoproceso y monohilo se tiene al MS-DOS. Los sistemas UNIX genéricos se pueden definir como sistemas multiproceso con un único hilo por proceso.

Casos de S.O. con hilos

- Dentro de los sistemas monoproceso multihilo se puede incluir a las máquinas virtuales Java. Los sistemas operativos más modernos se agrupan dentro de la categoría de multiproceso multihilo (W2003 y Solaris entre otros).

Ventajas de los hilos

- Respuesta: Una aplicación multihilo puede continuar ejecutándose incluso si una parte de ella está bloqueada o está realizando una tarea larga, incrementando de este modo el nivel de respuesta frente al usuario.

Ventajas de los hilos

- Compartición de recursos: Los hilos por defecto comparten la memoria y otros recursos del proceso al que pertenecen por lo que no hay que utilizar ningún mecanismo especial de compartición entre procesos. Esto permite que la aplicación pueda tener distintos hilos de ejecución con la misma memoria. Además los hilos aumentan la eficiencia de la comunicación entre programas en ejecución ya que pueden comunicarse sin la intervención del núcleo del sistema.

Ventajas de los hilos

- Economía: Reservar memoria y recursos en la creación de los procesos es costosa. Como los hilos comparten los recursos del proceso al que pertenecen, resulta más económico crear hilos nuevos y alternarlos en el procesador que hacer lo mismo con procesos. Aunque no existen medidas definitivas al respecto, en Solaris crear un proceso es unas 30 veces más lento que crear un hilo, y realizar un cambio de estado o contexto entre procesos es 5 veces más lento que hacerlo entre hilos.

Los hilos dentro de una misma aplicación comparten

- -Código (instrucciones)
- -Variables Globales
- -Archivos y dispositivos abiertos

Recursos no compartidos entre hilos:

- -Contador de programa (cada hilo puede ejecutar una sección distinta de código)
- -Registros de CPU
- -Pila para las variables locales.
- -Estado: Distintos hilos pueden estar en ejecución, listos o bloqueados esperando algún evento.

Estados de un Hilo

- -Los principales estados de un hilo son: ejecución, listo y bloqueado y hay 4 operaciones básicas relacionadas con el cambio de estado de los hilos:
- -**Creación:** En general cuando se crea un nuevo proceso se crea también un hilo para ese proceso. Posteriormente ese hilo puede crear nuevos hilos dándoles un apuntador de instrucción y algunos argumentos. Ese hilo se colocará en la cola de hilos listos.

Estados de un Hilo

- **Bloqueo**: Cuando un hilo debe esperar por un suceso, se le bloquea guardando sus registros. Así el procesador pasará a ejecutar otro hilo preparado.
- **Desbloqueo:** Cuando se produce el sucedo por el que un hilo se bloqueó pasa a la cola de listos.
- **Terminación**: Cuando un hilo finaliza, se libera su contexto y sus pilas.

Procesos e hilos

- Los hilos operan, en muchos sentidos, igual que los procesos:
- -Pueden estar en uno o varios estados: listo, bloqueado, en ejecución, terminado.
- -También comparten la CPU.
- -Solo hay un hilo activo (en ejecución) en un instante dado.
- -Un hilo dentro de un proceso se ejecuta secuencialmente.
- -Cada hilo tiene su propia pila y contador de programa.
- Puede crear sus propios hilos hijos.

Diferencia con los procesos

- Los hilos no son independientes entre sí. Como todos los hilos pueden acceder a todas las direcciones de la tarea, un hilo puede leer la pila de cualquier otro hilo o escribir sobre ella. Aunque pueda parecer lo contrario la protección no es necesaria ya que el diseño de una tarea con múltiples hilos tiene que ser de un único usuario.

Ventajas de los hilos sobre los procesos

- -Se tarda mucho menos tiempo en crear un nuevo hilo en un proceso existente que en crear un nuevo proceso.
- -Se tarda mucho menos tiempo en terminar un hilo que un proceso.
- -Se tarda mucho menos tiempo en conmutar entre hilos de un mismo proceso que entre procesos.
- -Los hilos hacen más rápida la comunicación entre procesos, ya que al compartir memoria y recursos se pueden comunicar entre si, sin invocar el núcleo del SO.

Tarea

- Investigar sobre las categorías de hilos
  - - Hilos a nivel de usuario
    - Ventajas
    - Desventajas
  - - Hilos a nivel de núcleo
    - Ventajas
    - Desventajas
- Investigar sobre el cambio de Contexto
- Investigar sobre la familia de exec y realizar los ejemplos de cada uno