UNIDAD 1 PARTE 2

Tabla de Contenidos

Tema 2: Procesos en un Sistema Operativo

Requerimientos

Todos los entornos de multiprogramación en los SO (ya sean los que soporten un usuario como Windows o miles como z/OS de IBM) tienen que cumplir los siguientes requerimientos:

- Debe ser capaz de intercalar Procesos (P) múltiples para maximizar el uso del procesador (μP ó nP) (con un $T_{respuesta}$ razonable)
- El SO debe reservar recursos a los procesos conforme a políticas definidas (prioridades), evitando los *deadlocks* o ínterbloqueos.
- El SO debe permitir la comunicación interproceso y la creación de los mismos por los usuarios.

`DEADLOCK o Interbloqueo: Cuando el ProcA espera a que el ProcB acabe una operación y, el ProcB también espera a que el ProcA acabe. Produciéndose un bloque mutuo`

2.1 Concepto de proceso: Estados de un proceso. Modelos de 5 estados de procesos.

Algunos conceptos previos:

- Existen una serie de recursos HW como el procesador, RAM, I/O, timers, HD,... $\Rightarrow$ Disponibles para las aplicaciones.
- El nP va cambiando las aplicaciones que procesa.
- El nP y los dispositivos I/O son utilizados eficentemente.

Proceso:

- Un programa en ejecución.
- Una instancia de un programa en ejecución.
- Puede ser asignada y ejecutada en un nP.
- Una unidad de actividad caracterizada por la ejecución de una secuencia de instrucciones, estado actual y con un conjunto de R (recursos) asociados,

2.1 Concepto de proceso: Estados de un proceso. Modelos de 5 estados de procesos.

Elementos esenciales del Proceso:

- Código de Programa: compartido con otros procesos que están ejecutando el mismo programa.
- Conjunto de datos: asociado a ese código.

PCB:

- Todos los procesos tienen una estructura de datos llamada PCB (Bloque de Control de Proceso) y es gestionada por el SO.
- Los Procesos consisten en código de programa asociado al PCB.
- Intercambio entre procesos guardando su información $\Rightarrow$ Dispatcher

Dispatcher: lo veremos después. Es el que "despacha" los procesos de la cola a la CPU y de la CPU a la cola.

2.1 Concepto de proceso: Estados de un proceso. Modelos de 5 estados de procesos.

Mientras el P está en ejecución:

- Identificador
- Estado
- Prioridad
- PC (Program Counter)
- Punteros a memoria
- Contexto
- Información de estado de I/O
- Accounting (clock, time used, time limits, ...)

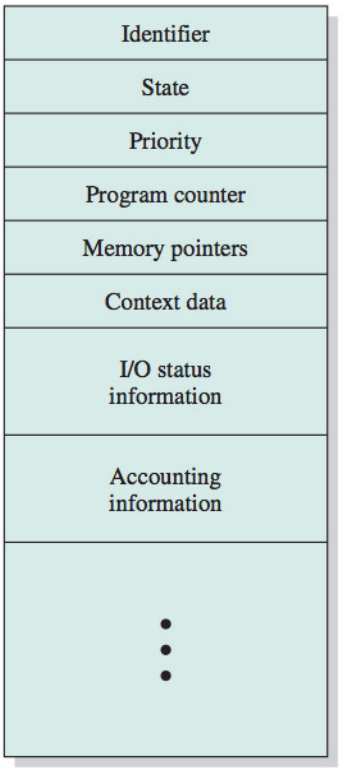


Fig. 1 : PCB

2.1 Concepto de proceso: Estados de un proceso. Modelos de 5 estados de procesos.

Cada proceso va a ocupar un espacio en memoria que es finita.

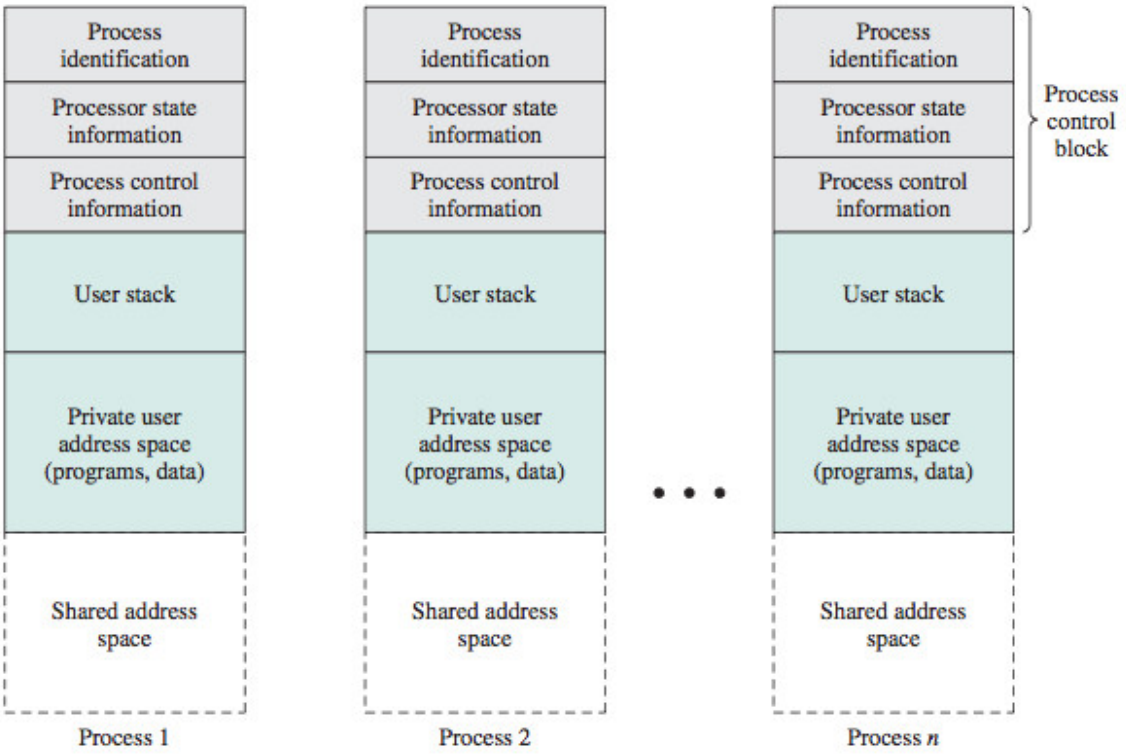


Fig. 2 : Procesos y sus PCBs

2.1 Concepto de proceso: Estados de un proceso. Modelos de 5 estados de procesos.

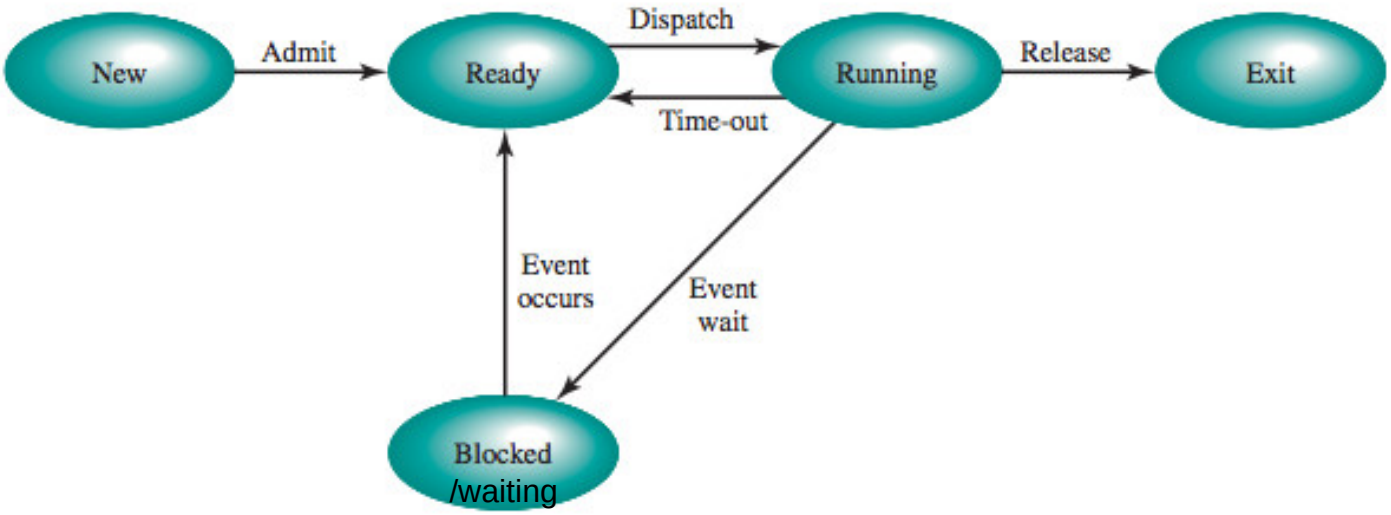


Fig. 3 : Modelo de 5 estados

2.1 Concepto de proceso: Estados de un proceso. Modelos de 5 estados de procesos.

Los *cinco* estados de un Proceso son los siguientes:

- **Running:** en ejecución (con 1P, al menos uno)
- **Ready:** preparado para ejecutarse cuando tenga oportunidad.
- **Blocked/Waiting:** un P no puede ejecutarse hasta que ocurra un evento (operación I/O completa).
- **New:** un P nuevo no se carga en MP hasta que el PCB se haya creado y se ejecutará cuando el SO lo admita.
- **Exit:** un P es liberado de lista de P's ejecutables por el SO (también por que sea abortado o parado por otras razones).

2.1 Concepto de proceso: Estados de un proceso. Modelos de 5 estados de procesos.

Transiciones entre estados:

- **Null→New:** Proceso creado.
- **New→Ready:** Cuando el SO decide que el P tiene todo listo.
- **Ready→Running**: Dispatcher elige el Proceso que entrará en ejecucicón.
- **Running→Exit**: Cualaquier causa: (tarea finalizada, memoria insuficiente, abortado por otro proceso, etc.)
- **Running→Ready:** Dispatcher lo envía a la cola por agotamiento de tiempo.
- **Running→Blocked**: El proceso se bloquea por una operación I/O.
- **Blocked→Ready:** Finaliza la operación I/O y se envía a la cola de Ready.
- **Ready→Exit:** El padre puede terminar sus hijos.
- **Blocked→Exit:** Si un proceso pasa mucho tiempo bloqueado, el SO lo termina.

2.1 Concepto de proceso: Estados de un proceso. Modelos de 5 estados de procesos.

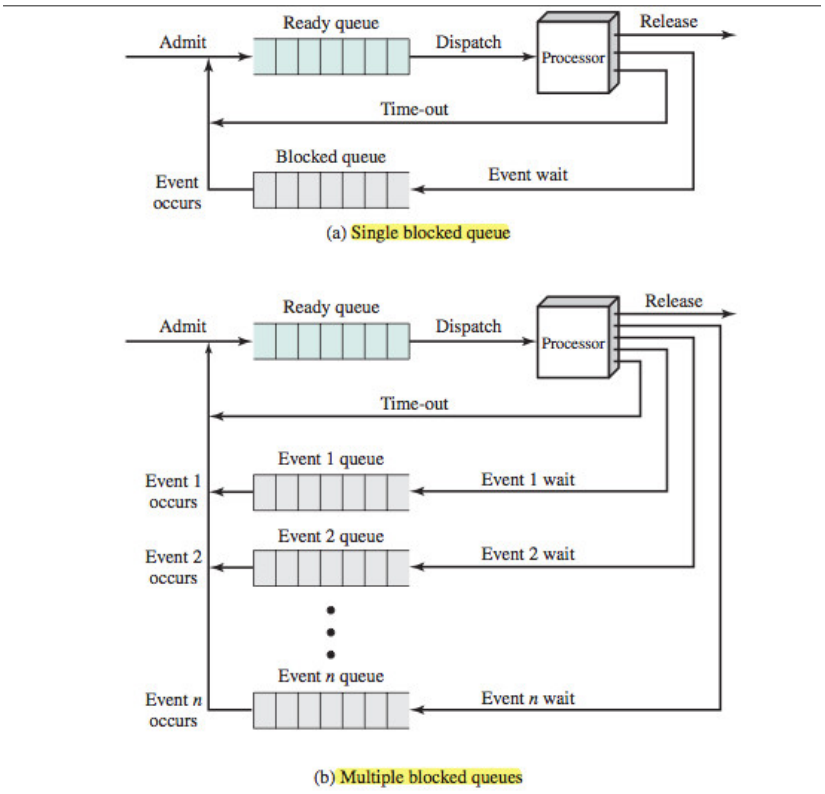


Fig. 4 : (a) cola de procesos única; (b) cola múltiple de procesos

2.2 Control de procesos. Modos de ejecución. Creación de procesos.

Modos de Ejecución:

- Modo Usuario (user mode): los programas se ejecutan típicamente en este modo.
- Modo Kernel (system mode, control mode, kernel mode): realiza importantes funciones del sistema.

Objetivo

Es necesario proteger el SO: los PCBs, tablas de sistema, etc., de la interferencia de programas de usuario.

2.2 Control de procesos. Modos de ejecución. Creación de procesos.

Funciones del modo kernel:

***Gestión de Procesos**

- Creación y finalización
- Despacho y planificación de procesos
- Cambio de un proceso a otro
- Sincronización de proceos y soporte para comunicación entre procesos.
- Gestión de PCBs

*** Gestión de Memoria**
- Asignación de espacio de direcciones de memoria a los procesos
- Swapping
- Gestión de páginas y segmentos

***Gestión I/O**
- Gestión de buffer
- Asignación de canales de I/O y de dispositivos a procesos

***Funciones de soporte**
- Manejo de interrupciones
- Auditoría
- Monitorización

2.2 Control de procesos. Modos de ejecución. Creación de procesos.

Eventos por los que se crea un proceso:

- Nuevo batch job: cuando el SO toma un nuevo trabajo, leyendo la secuencia de comandos de control de trabajos.
- Log-on interactivo: un usuario se loquea en un terminal.
- Creado por el SO para un servicio: el SO crea un proceso para realizar una función para un programa de usuario sin que el usuario tenga que esperar (p.e. un proceso de impresión).
- Generación por otro proceso existente: un programa de usuario pude generar la creación de un número de procesos.

2.2 Control de procesos. Modos de ejecución. Creación de procesos.

RAZONES POR LAS QUE UN PROCESO FINALIZA (ESTADO "EXIT")

Normal completion	The process executes an OS service call to indicate that it has completed running.
Time limit exceeded	The process has run longer than the specified total time limit. There are a number of possibilities for the type of time that is measured. These include total elapsed time ("wall clock time"), amount of time spent executing, and, in the case of an interactive process, the amount of time since the user last provided any input.
Memory unavailable	The process requires more memory than the system can provide.
Bounds violation	The process tries to access a memory location that it is not allowed to access.
Protection error	The process attempts to use a resource such as a file that it is not allowed to use, or it tries to use it in an improper fashion, such as writing to a read-only file.
Arithmetic error	The process tries a prohibited computation, such as division by zero, or tries to store numbers larger than the hardware can accommodate.
Time overrun	The process has waited longer than a specified maximum for a certain event to occur.
I/O failure	An error occurs during input or output, such as inability to find a file, failure to read or write after a specified maximum number of tries (when, for example, a defective area is encountered on a tape), or invalid operation (such as reading from the line printer).
Invalid instruction	The process attempts to execute a nonexistent instruction (often a result of branching into a data area and attempting to execute the data).
Privileged instruction	The process attempts to use an instruction reserved for the operating system.
Data misuse	A piece of data is of the wrong type or is not initialized.
Operator or OS intervention	For some reason, the operator or the operating system has terminated the process (e.g., if a deadlock exists).
Parent termination	When a parent terminates, the operating system may automatically terminate all of the offspring of that parent.
Parent request	A parent process typically has the authority to terminate any of its offspring.

Fig. 6 : Eventos por los que termina un proceso

2.2 Control de procesos. Modos de ejecución. Creación de procesos.

La *Creación de un Proceso* se lleva a cabo de la siguiente forma:

1. Asignar un identificado de proceso único al nuevo proceso
2. Reservar espacio para el proceso: el SO debe conocer cuánto espacio requiere para el usuario (programa y datos).
3. Inicializar el PCB: se inicializan los descriptores, memoria, IDs, atributos por defecto del proceso que lo creó.
   - El proceso se inicializa en estado *Ready*.
   - Prioridad $\rightarrow$ (suele tener una prioridad baja, a menos que se le especifique una prioridad más alta).
   - No hay recursos (I/O, ficheros) asignados por defecto (a menos que vengan heredados por el padre).

2.2 Control de procesos. Modos de ejecución. Creación de procesos.

4. Configurar enlaces: por ejemplo, si el SO mantiene una cola de planificación, el nuevo proceso debe ponerse en esa lista.
5. Crear o expandir otras estructuras de datos: el SO debe tener un fichero de auditoría de cada proceso.

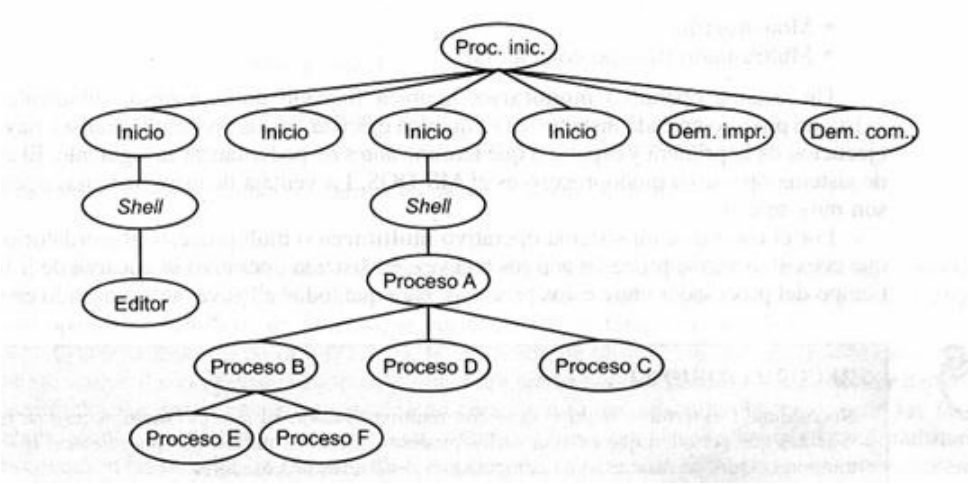


Fig. 7 : Jerarquía de procesos en Linux

2.3 Administración de Procesos en Unix/Linux

2.3 Administración de Procesos en Unix/Linux

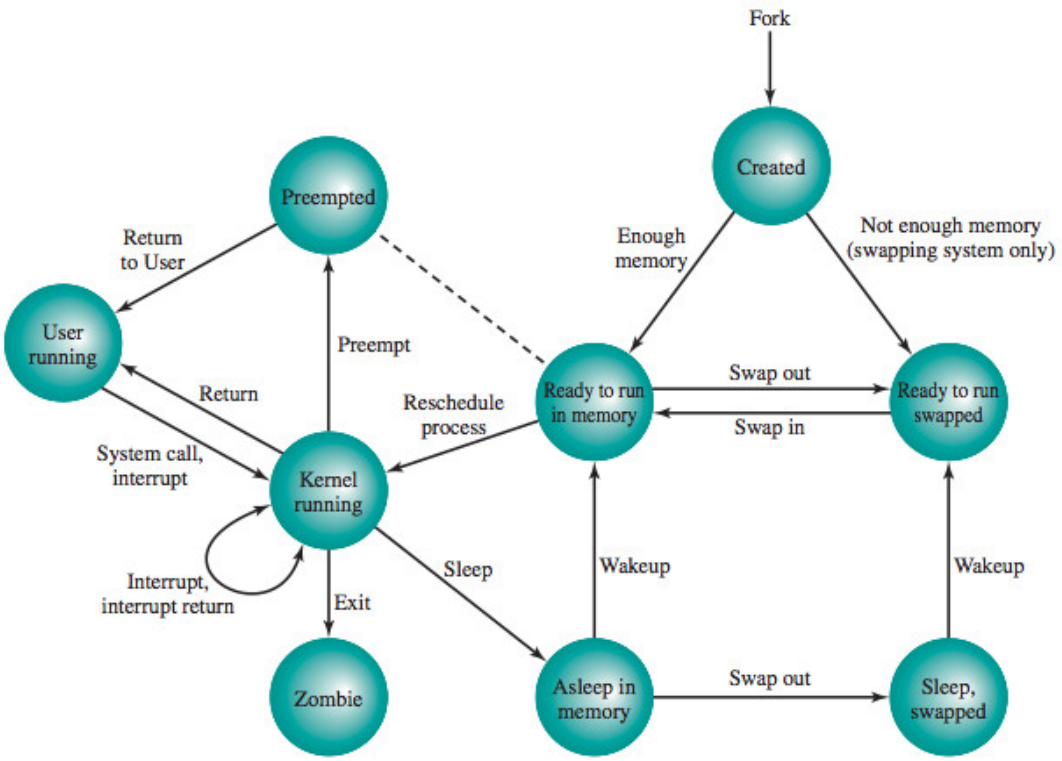
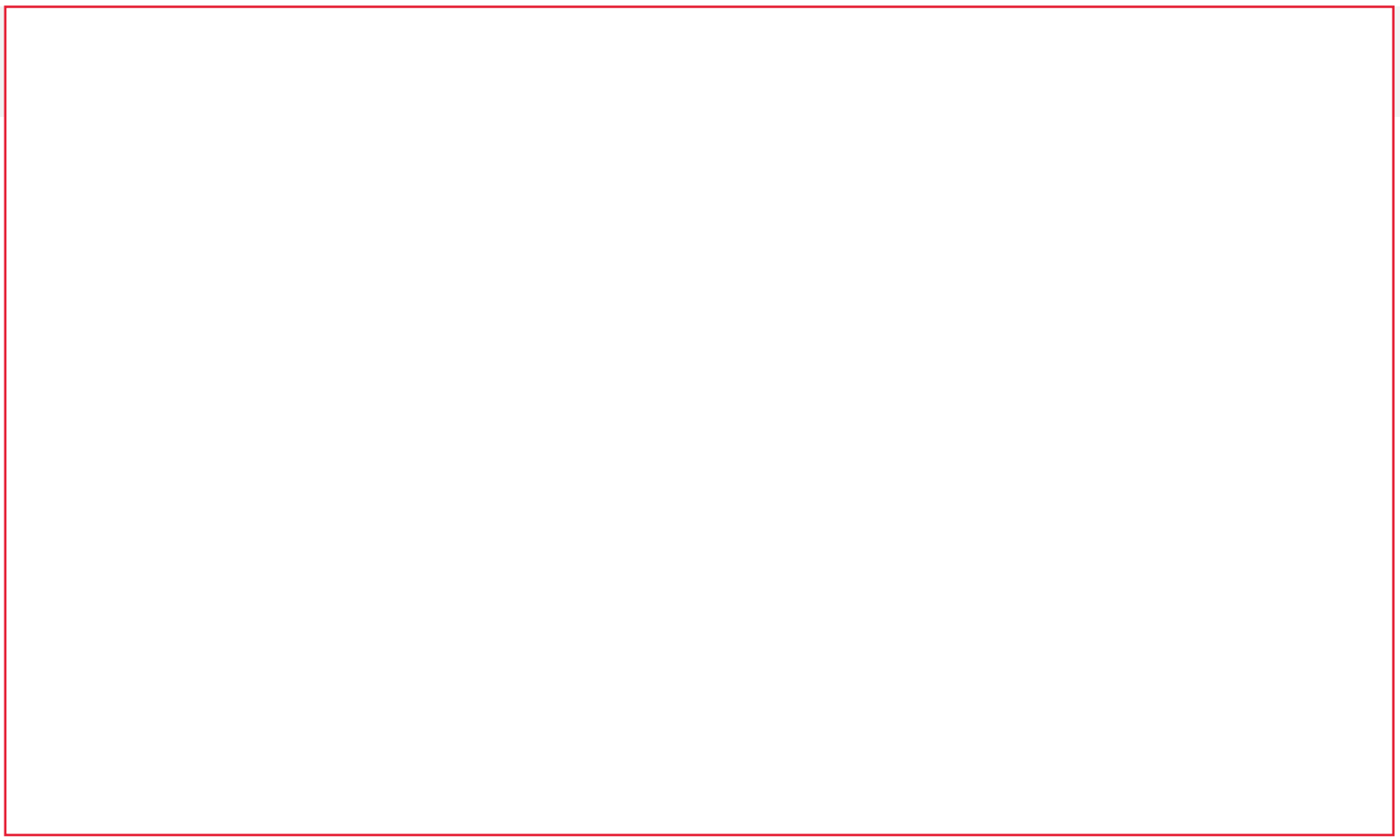


Fig. 8 : Llamada fork en Unix



2.3 Administración de Procesos en Unix/Linux

En Linux un proceso (o tarea) es representada por una estructura de datos llamada *task_struct* y que contiene las siguientes categorías de información:

Linux Tasks

- **State:** o estado de ejecución del proceso (executing, ready, suspended, stopped, zombie).
- **Scheduling information:** información necesaria para la planificación de procesos (procesos en Tiempo Real son planificados antes de los normales: aplicaciones de video/audio, sensores, etc.).
- **Indentifiers:** cada proceso tiene un identificador único.
- **Interprocess communication:** Linux soporta mecanismos de IPC (heredados de Unix).
- **Links:** cada proceso incluye un link a el proceso padre (parent_process) y a sus procesos hijos.
- **Times and Timers:** Tiempo de creación y tiempo consumido en el procesador.

2.3 Administración de Procesos en Unix/Linux

Linux Tasks

- **File System:** Incluye punteros a los ficheros abiertos por el proceso y a los directorios.
- **Address space:** espacio de direcciones virtuales asignadas al proceso.
- **Processor-specific context:** registros y pila de contexto del proceso.
- **Running:** si está ejecutándose (Runnning) o preparado (Ready).
- **Interruptible:** proceso bloqueado (Blocked) esperando un evento (recurso, I/O, señal de otro proceso).
- **Uninterruptible:** proceso bloqueado y en espera de condiciones HW que le permitan seguir.
- **Stopped:** Proceso parado y a la espera de que lo ejecuten.
- **Zombie:** Proceso terminado aunque sigue en la task_structure de la tabla de procesos.

2.3 Administración de Procesos en Unix/Linux

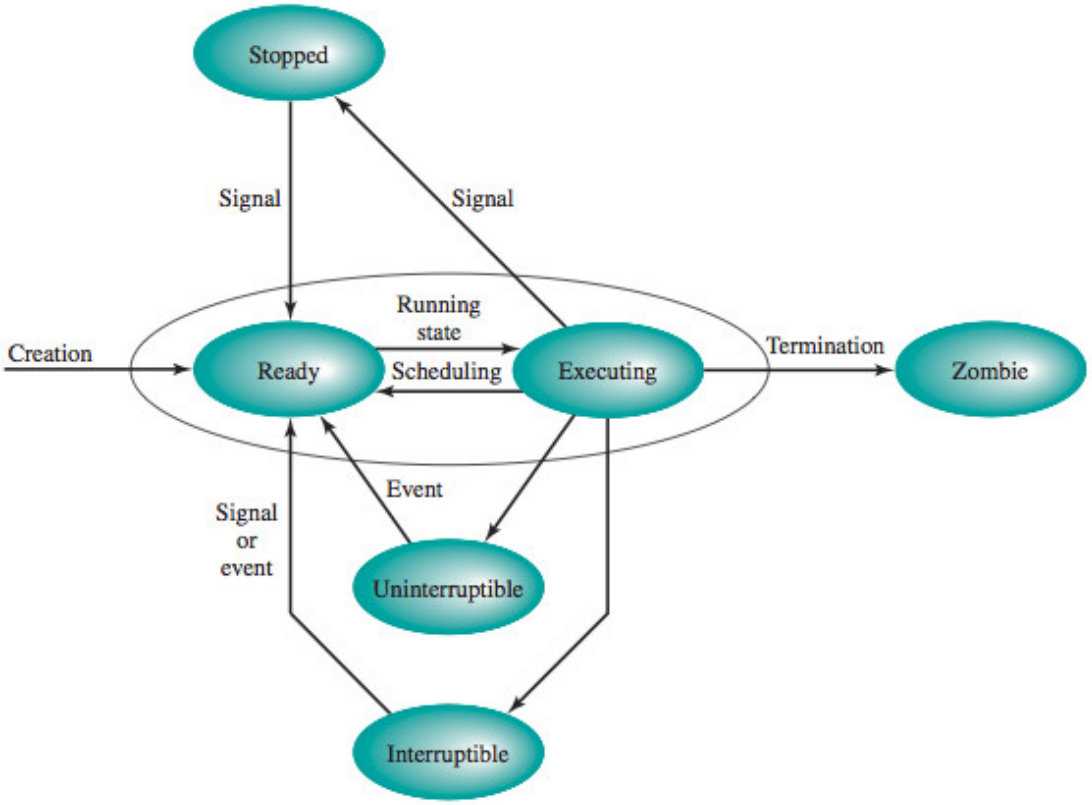


Fig. 9 : Diagrama de estados de procesos en Linux

2.4 Procesos e hilos

Los procesos tienen las siguientes características:

1. **Propiedad de los procesos:**
   - incluye un espacio de MV que alberga la imagen del proceso (programa, datos, pila, PCB).
   - puede tener asignado recursos, memoria, dispositivos, ficheros.
2. **Planificación/ejecución:** La ejecución de un proceso es intercalada con otros.
   - Dispatcher: el encargado de elegir los Procesos a la CPU de la cola.
   - Prioridad: nivel de ejecución

Ver prioridades Win, OSX y Linux: http://bit.ly/2xXHsJ2

Definición:

Un hilo o *thread* es un proceso ligero, el cual va a ser menos costoso de gestionar que un proceso. Los hilos se ejecutan dentro de un proceso, el cual puede tener uno o varios.

2.4 Procesos e hilos

Multithreading

- Es la capacidad de que un SO soporte múltiples, y de forma concurrente, varios hilos en un mismo proceso.

Example

- MSDOS tiene un sólo proceso de usuario y un sólo thread.
- La JVM (Java Run Time) es un ejemplo de un proceso con múltiples hilos o threads.

Hilos y los SO modernos

Generalmente, en los sistemas Unix/Linux, Windows, MacOS, IBM, etc., implementan sistemas de *multithreading*

2.4 Procesos e hilos

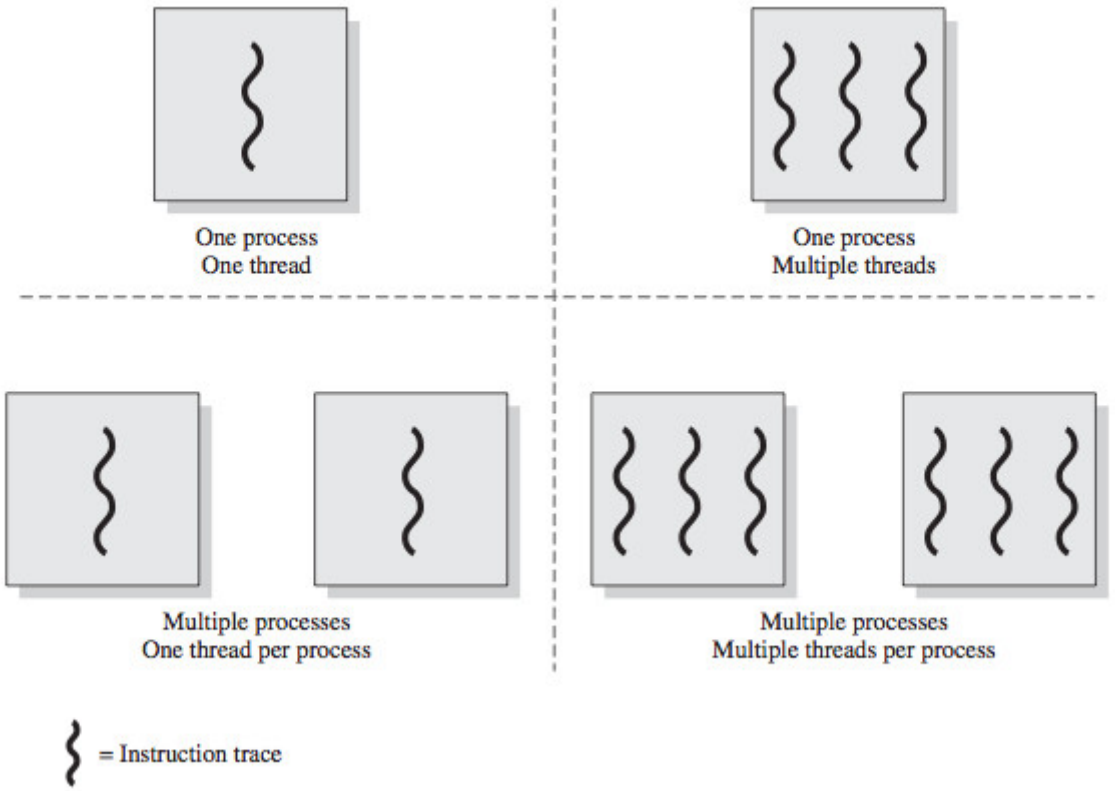


Fig. 10 : Multithreading

2.4 Procesos e hilos

Dentro de un proceso, puede haber uno o más **hilos** con lo siguiente:

- Estado de ejecución (Running, Ready, etc.).
- Un contexto de hilo salvado (pero no en ejecución).
- Pila de ejecuión.
- Variables locales.
- Memoria y recursos compartidos con otros hilos.

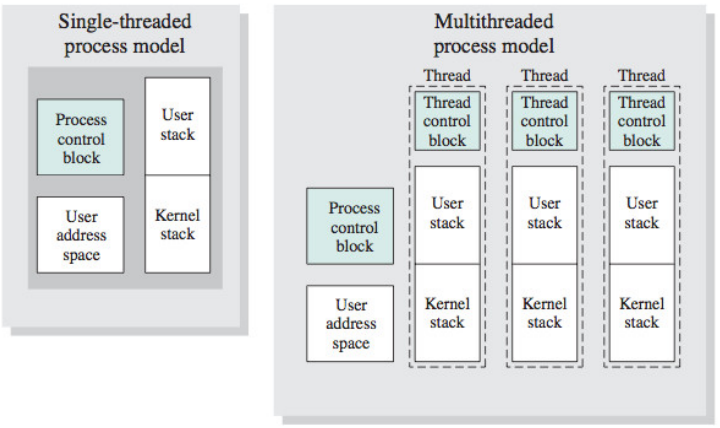


Fig. 11 : Hilo en único proceso y en varios

2.4 Procesos e hilos

Características:

- Todos los hilos de un proceso comparten el estado, los recursos, el espacio de memoria y tienen acceso a los mismos datos.
- Si un hilo altera un dato en memoria, los otros hilos ven el resultado.
- Si un hilo abre un fichero con privilegios de sólo lectura, los otros también pueden leerlo al mismo tiempo.

2.4 Procesos e hilos

Ventajas de los hilos:

- Es mucho menos costoso crear un nuevo hilo en un proceso existente que crear un proceso (como x10).
- Es mucho menos costoso terminar un hilo que un proceso.
- De igual forma, cambiar entre dos hilos es mucho menos costoso que cambiar entre procesos.
- Mejora la comunicación entre los distintos programas en ejecución $\rightarrow$ En la comunicación de procesos se requiere la intervención del kernel (mayor coste).

Examples

- Un servidor de ficheros: responde a muchas peticiones $\rightarrow$ muchos hilos creados y destruidos en un corto espacio de tiempo.
- Si hubiera más de un procesador $\rightarrow$ el proceso podría ejecutarse simultáneamente.
- Es más rápido utilizar hilos para procesos que comparten memoria.

2.4 Procesos e hilos

Examples

- Tareas en (foreground y background): una hoja de cálculo → un hilo puede visualizar menús y recibir datos de entrada, mientras otro hilo ejecuta comandos de usuario y actualiza la hoja → mayor sensación de rapidez.
- Proceso asíncrono: hilo que guarda la información de un procesador de textos de RAM a disco cada minuto.
- Velocidad de ejecución: un proceso multihilo puede realizar una tarea match mientras lee el siguiente trabajo en el dispositivo.
- Estructura de programa modular: programa que integran una gran variedad de actividades, dispositivos destinos y fuentes.

2.4 Procesos e hilos

Suspensión y terminación:

- Si un hilo se suspende (al compartir memoria, etc.) $\rightarrow$ se suspenden todos los hilos del proceso.
- Si un proceso termina $\rightarrow$ terminan también los hilos dentro de ese proceso.

Estados de un hilo:

- **Spawn**(engendrar): creación de hilo.
- **Block:** cuando un hilo espera un evento se bloquea (salva registros, PC, stack) $\rightarrow$ el SO ejecuta otro hilo de otro o el mismo proceso.
- **Unblock:** cuando un hilo se desbloquea pasa a la cola de READY.
- **Finish:** Un hilo termina y su información es desasignada.

2.4 Procesos e hilos

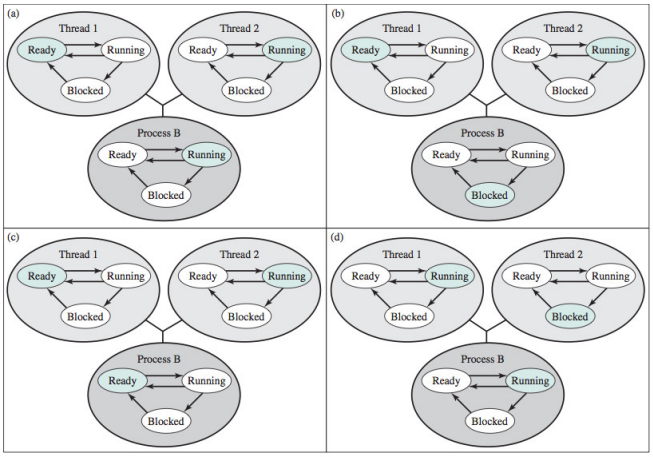


Fig. 12 : Relaciones de estados de hilos y procesos

Threads: Processes	Description	Example Systems
1:1	Each thread of execution is a unique process with its own address space and resources.	Traditional UNIX implementations
M:1	A process defines an address space and dynamic resource ownership. Multiple threads may be created and executed within that process.	Windows NT, Solaris, Linux, OS/2, OS/390, MACH
1:M	A thread may migrate from one process environment to another. This allows a thread to be easily moved among distinct systems.	Ra (Clouds), Emerald
M:N	Combines attributes of M:1 and 1:M cases.	TRIX

Hilos:Procesos	Descripción	Sistema Ejemplo
1:1	Cada hilo de ejecución es un único proceso con su propio espacio de memoria y recursos	Unix
M:1	Un proceso define un espacio de direcciones de memoria y recursos propios. Múltiples hilos son creados y ejecutados dentro del proceso.	Win NT, Solaris, Linux, OS/2, Mach
1:M	Un hilo migra de un proceso a otro. Permite que un hilo se mueva en distintos sistemas.	Sistemas Cloud.