Sistemas Operativos

Martín Silva Alejandro Mansilla

Licenciatura en Ciencias de la Computación

Agosto 2018

Sistemas Operativos

Procesos

- "Un proceso es un programa en ejecución". Saltzer (1966).

  El término "proceso" se refiere al conjunto de ejecución de instrucciones de máquina.

Programa	Proceso
Es estático	Es dinámico
No tiene contador de programa	Tiene contador de programa
Existe desde que se edita hasta que se borra	Su ciclo de vida comprende desde que se lo "dispara" hasta que termina

UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO

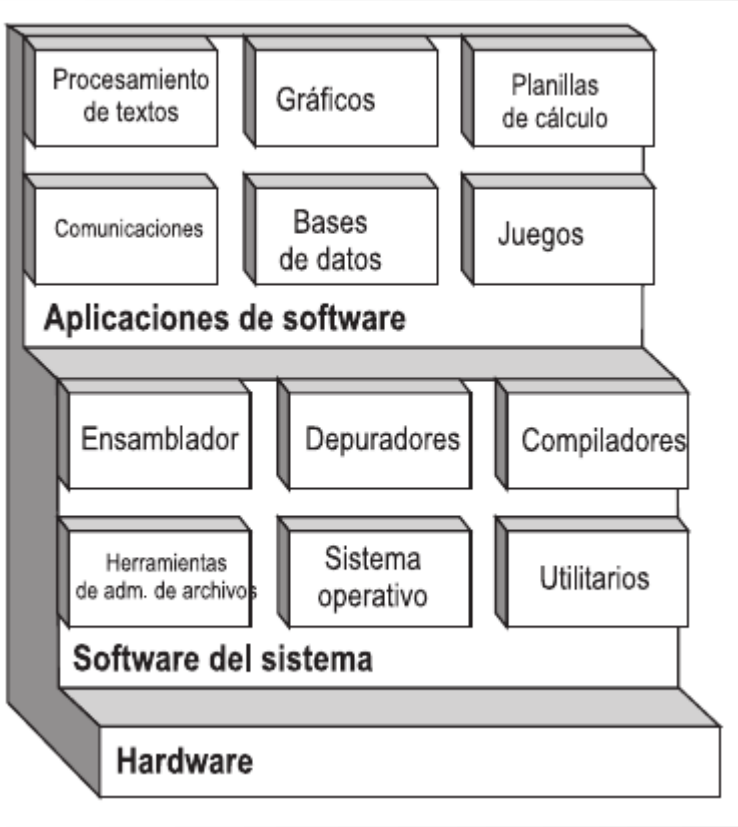
Sistemas Operativos

Programas

- **Aplicación o software de aplicación**: programas o grupos de programas diseñados para ser utilizados directamente por los usuarios (también llamados usuarios finales).

- **Software de sistemas**: son los programas que interactúan de forma más directa con el sistema operativo, tales como los compiladores, depuradores y ensambladores.

En la figura podemos observar cómo se apoyan las aplicaciones sobre el software de sistema porque, figurativamente hablando, no pueden ejecutar sin él. Pero para nuestro estudio a todos les llamaremos programas.



UNCUYO
UNIVERSIDAD NACIONAL DE CUYO

Sistemas Operativos

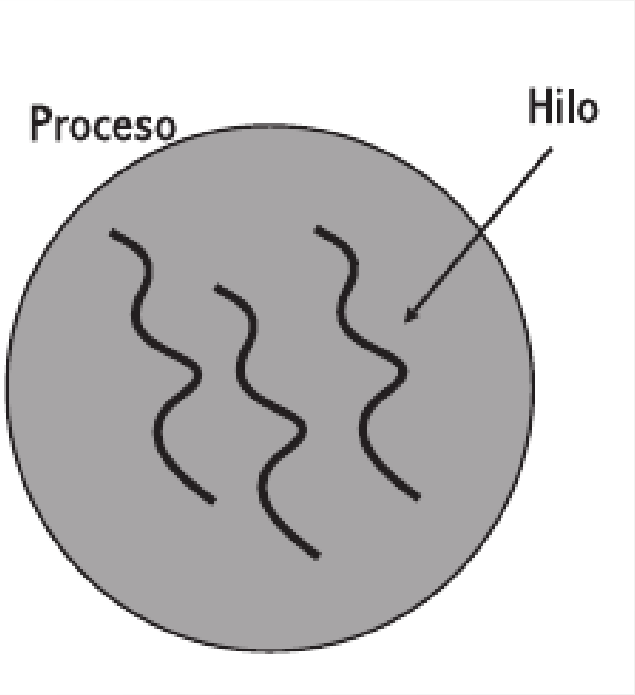
Hilos

- Asociados a un proceso encontramos su código, sus datos, los recursos operativos asignados a él y uno o más “hilos” (threads) de ejecución.

Un hilo es un flujo de ejecución a lo largo del código del proceso, con su propio contador de programa, registros y pila. Pueden compartir el mismo espacio de direcciones, es decir que pueden examinar y modificar las mismas variables.

Los sistemas operativos pueden clasificarse en tres tipos de sistemas:

- De un único proceso con un único hilo.
- De varios procesos con un único hilo.
- De múltiples procesos con varios hilos.



UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO

Estados de un proceso

Decíamos con anterioridad que un proceso es dinámico; a medida que se ejecuta, cambia su estado. El estado de un proceso está definido en parte por la actividad actual de ese proceso. Un proceso puede estar, mínimamente, en uno de los tres siguientes estados:

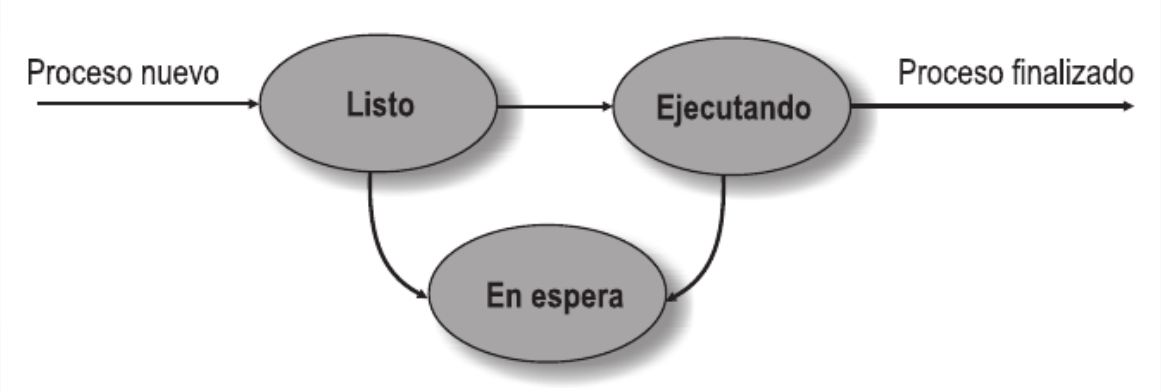


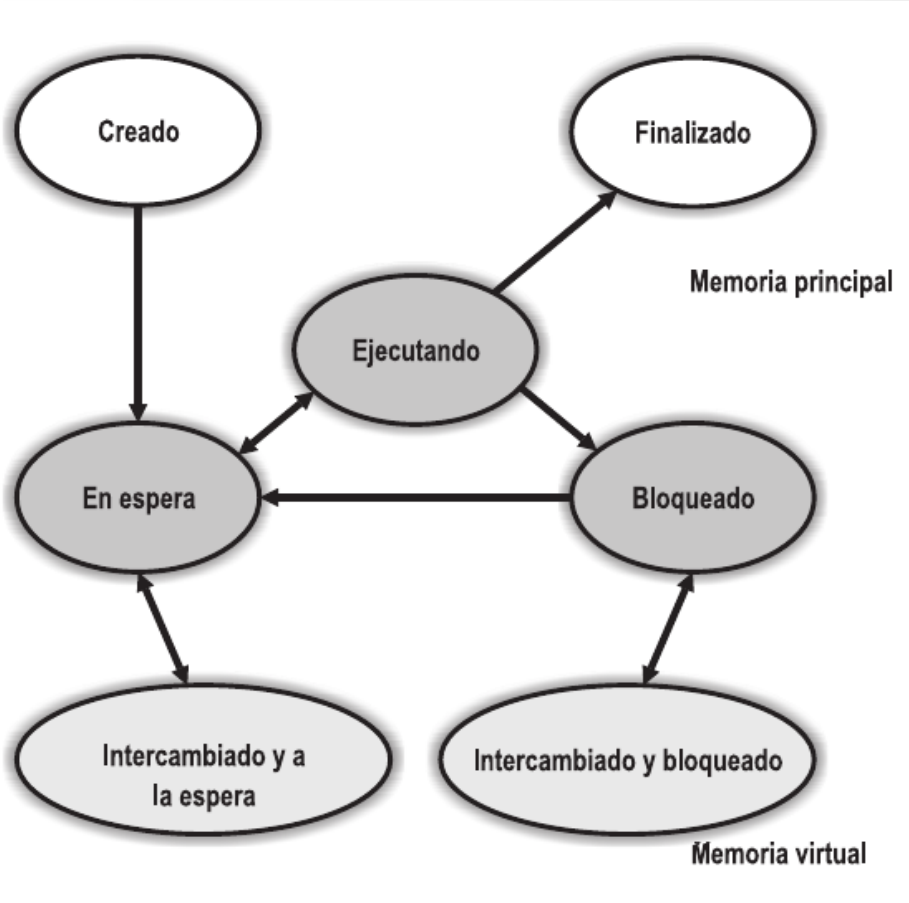
Diagrama de transición de estados en Multics

- **Ejecutándose** (*running*): un proceso que está ejecutando en una CPU. Si un sistema tiene n CPU puede tener como máximo n procesos en ese estado.
- **Bloqueado** (*blocked*): el proceso está esperando que ocurra algún evento, como por ejemplo la finalización de una operación de entrada o salida.
- **Listo** (*ready*): un proceso que no está asignado a la CPU pero está listo para ejecutar. Un proceso listo podría ejecutar si se lo asigna a una CPU.

Estados de un proceso

Según Stallings, considerando a los sistemas operativos modernos, deberíamos defi

- **Nuevo** (*new*): también llamado "creado" justamente porque el proceso se está creando, espera admisión al estado "listo".
- **Listo** (*ready*): también llamado "esperando", ha sido cargado en memoria principal y espera que el despachador lo ponga a ejecutar. Puede haber varios procesos en este estado.
- **Bloqueado** (*blocked*): un proceso que está esperando que ocurra un evento antes de que pueda continuar. Con frecuencia, este evento es la finalización de una operación de entrada o salida.
- **Terminado** (*terminated*): un proceso que ha detenido su ejecución pero el sistema operativo aún mantiene un registro de él (en UNIX, a estos procesos se les suele llamar "*zombie*").



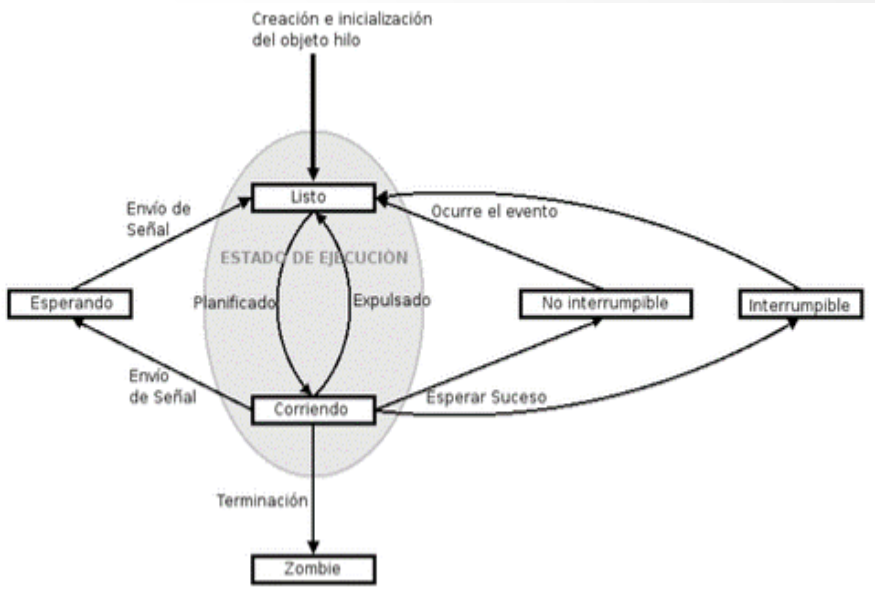
Podemos definir dos estados más en los sistemas que cuentan con memoria virtual:

- **Intercambiado y esperando**: también se lo denomina "suspendido y esperando", fue sacado de la memoria principal y almacenado en memoria secundaria.
- **Intercambiado y bloqueado**: o "suspendido y bloqueado"; un proceso que estaba bloqueado también puede pasar a residir en el área de *swap*.

UNCUYO
UNIVERSIDAD NACIONAL DE CUYO

Estados de un proceso en Linux

- **Uninterruptible sleep** (D): Espera initerrumpible, generalmente el proceso se encuentra esperando una operación de entrada/salida con algún dispositivo.
- **Interruptible sleep** (S): Espera interrumpible. El proceso se encuentra esperando a que se cumpla algún evento, por ejemplo, que el planificador de procesos del núcleo lo planifique para su ejecución.
- **Running** (R): Corriendo. El proceso se encuentra corriendo en el procesador.
- **Stopped** (T). Detenido. Un proceso que ha sido detenido mediane el envío de alguna señal.
- **Defunct** (Z - zombie). Proceso terminado pero cuyo padre aún sigue "vivo" y no ha capturado el estado de terminación del proceso hijo y, por consiguiente, no lo ha eliminado de la tabla de procesos del sistema.



PCB - Bloque de Control de Procesos

Cada proceso se representa en el sistema operativo mediante una estructura de datos llamada bloque de control de procesos (**PCB** de *Process Control Block*)

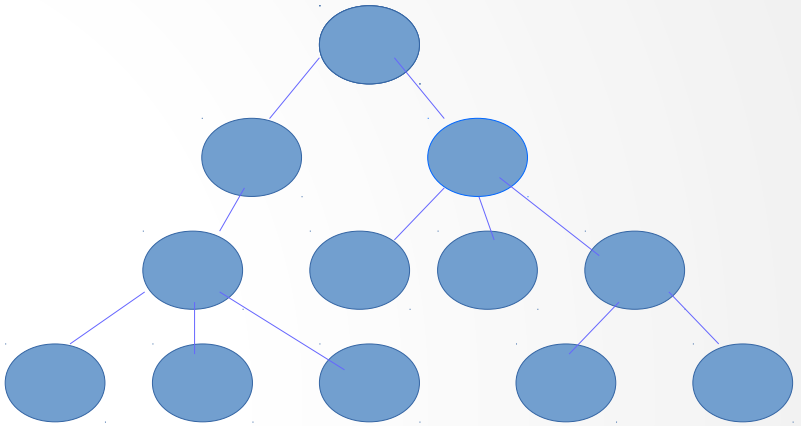
- Estado del proceso: nuevo, bloqueado, listo, etc.
- Contador de programa: indica la dirección de la siguiente instrucción que se ejecutará.
- Registros de CPU: el número y el tipo de los registros varían dependiendo de la arquitectura del computador (acumuladores, registros índice, punteros de pila, etc).
- Información de planificación de CPU: prioridad del proceso, punteros a colas de planificación.
- Información de gestión de memoria: valor de los registros de base y de límite, tablas de páginas o tablas de segmentos.
- Información contable: cantidad de tiempo de CPU, tiempo consumido, límites de tiempo.
- Información de estado de entrada o salida: lista de dispositivos de entrada o salida asignados, lista de archivos abiertos, etc.

Identificador del proceso
Estado
Contador de programa
Registros de CPU
Planificación de CPU
Gestión de memoria
Información contable
Estado de entrada o salida

UNCUYO
UNIVERSIDAD NACIONAL DE CUYO

Creación y terminación de procesos - Jerarquía

- En Unix, un proceso crea a otro a través de una llamada a sistema -**fork()**-. El creador es el proceso padre y el creado es el hijo. El proceso hijo puede crear otro(s) generándose una estructura de árbol con vértice en el padre: es una estructura jerárquica.
- En la ejecución ante la creación de un subproceso hay dos alternativas:
  - El padre continúa ejecutándose concurrentemente con sus hijos.
  - El padre espera que sus hijos hayan terminado para seguir la ejecución.



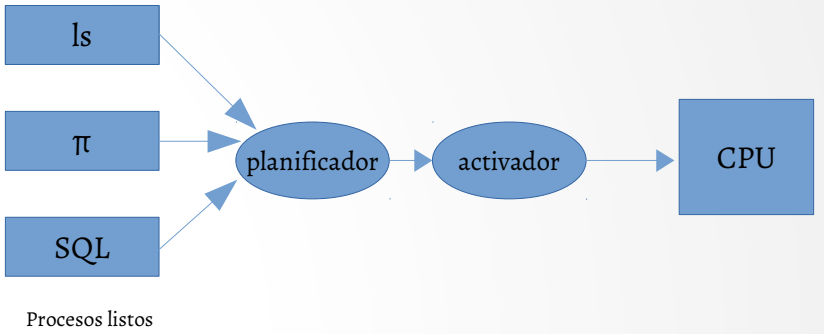
Árbol de procesos en sistemas tipo Unix

- Un proceso termina cuando ejecuta su última instrucción y solicita al sistema operativo que lo elimine.
- Con fork() los procesos se clonan. Ambos procesos continúan su ejecución con la instrucción siguiente al fork(). El código de retorno que recibe el hijo es 0, el del padre es el PID del hijo.
- Si el hijo invoca **execve()** se reemplaza con un programa nuevo.
- El padre puede esperar la finalización del hijo con **wait()**.
- Windows tiene el concepto de "trabajo" (*job*) para que un grupo de de procesos sean gerenciados como una unidad..

Sistemas Operativos

Planificador y activador

- El planificador (*scheduler*) es un código que forma parte del núcleo del sistema operativo.
- Entra en ejecución cada vez que se activa el sistema operativo y su misión es seleccionar el proceso que se ha de ejecutar a continuación.
- El activador (*dispatcher*) también forma parte del sistema operativo y su función es poner en ejecución el proceso seleccionado por el planificador.

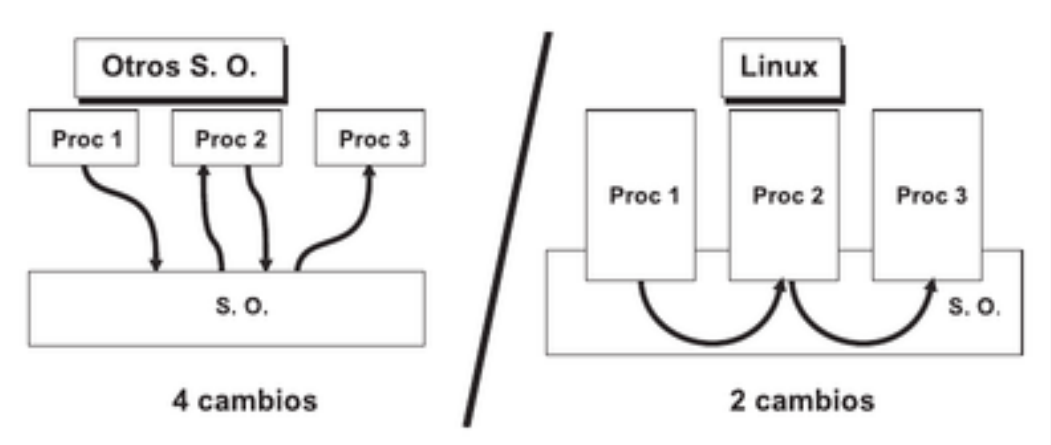


- La planificación puede ser dividida en tres fases: la de largo, la de medio y la de corto plazo:
  - La planificación a largo plazo determina qué trabajos o procesos pueden competir por los recursos del sistema. Provee al de mediano de un apropiado número de trabajos.
  - El planificador de mediano plazo o intercambiador (*swapper*) intercambia procesos entre la memoria RAM y un espacio en disco.
  - El planificador de corto plazo selecciona al proceso que será asignado a la CPU y el activador es el que efectivamente carga el proceso en la CPU y es también el que se encarga de pasar a modo usuario .

UNCUYO
UNIVERSIDAD NACIONAL DE CUYO

Cambio de contexto

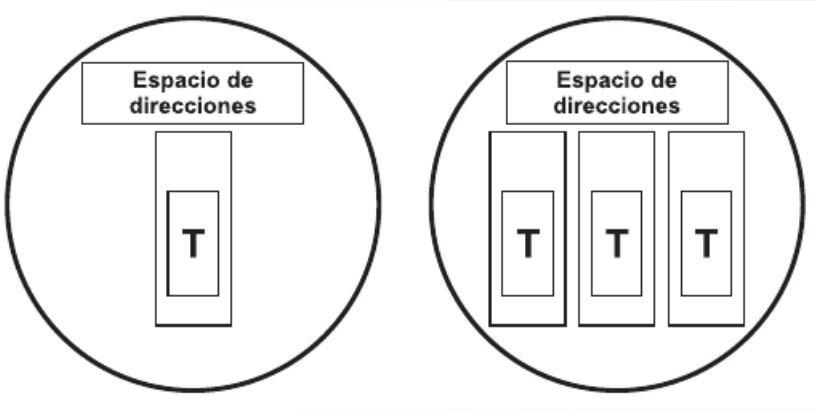
- La activación del sistema operativo se realiza mediante el mecanismo de las interrupciones. Cuando se produce una interrupción se realizan las dos operaciones siguientes:
  - Se salva el estado del procesador en el correspondiente PCB.
  - Se pasa a ejecutar la rutina de tratamiento de interrupción del sistema operativo.
- Llamaremos **cambio de contexto** (*context switch*) al conjunto de estas operaciones.
- El tiempo de conmutación de contexto es exclusivamente gasto extra (*overhead*), porque el sistema no realiza trabajo útil durante la conmutación.
- Al tiempo entre detener un proceso y comenzar a correr otro se le llama "latencia de activación" o *dispatch latency*.



Latencia: medida de una demora temporal experimentada en un sistema

Procesos ligeros, hilos o *threads*

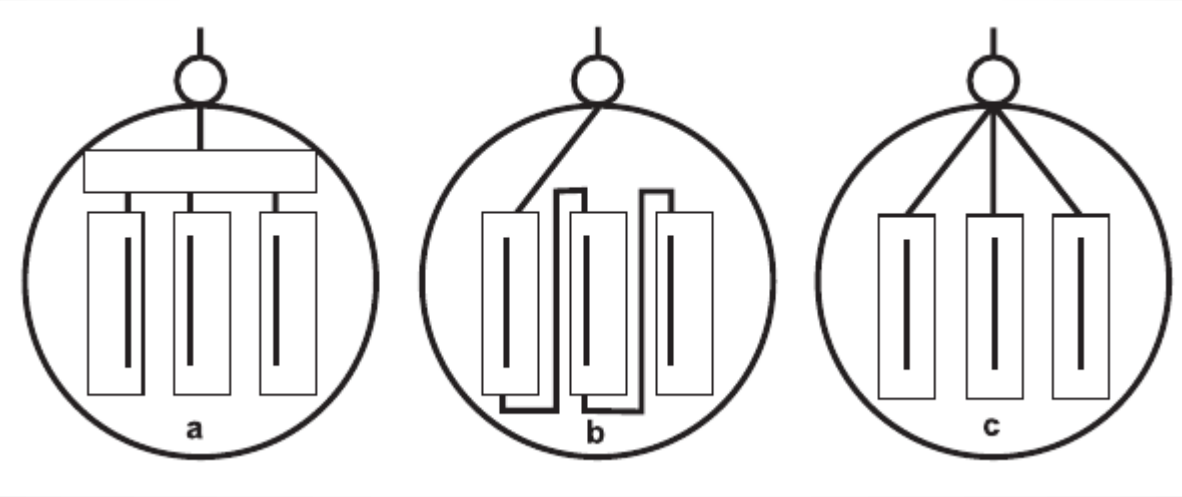
- Un hilo es un flujo de ejecución que comparte la imagen de memoria, y otras informaciones, con otros hilos.
- Los SO que pueden ejecutar varios procesos con múltiples hilos usan con más eficiencia la CPU. El sistema intercala la ejecución de procesos entre la CPU y la entrada/salida.



- Las motivaciones para el uso de hilos pueden ser:
  - La sobrecarga involucrada en la creación de un proceso es mayor que en la de un hilo.
  - La conmutación entre hilos del mismo proceso es más rápida que entre procesos.
  - Los hilos permiten que el paralelismo sea combinado con la ejecución secuencial y llamadas a sistema bloqueantes.
  - El compartir recursos puede lograrse de manera más eficiente y naturalmente entre hilos de un proceso que entre procesos, porque todos los hilos de un proceso comparten el mismo espacio de direcciones.

Modelos de organización de hilos

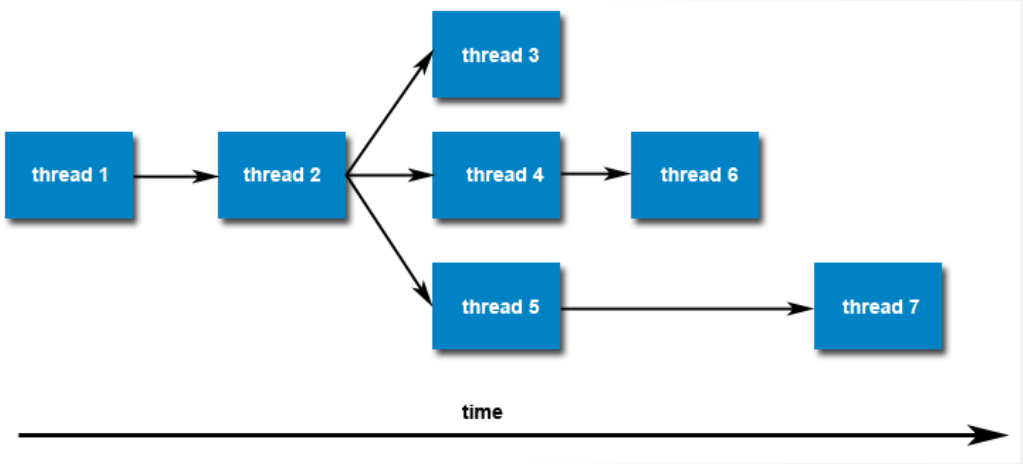
a) Modelo despachador-trabajadores: Un hilo despachador (único) y varios hilos trabajadores. El hilo despachador acepta pedidos de clientes y, luego de examinar el pedido, despacha el mismo a uno de los hilos trabajadores libres para su procesamiento.
b) Modelo pipeline: Útila para aplicaciones basadas en el modelo productor-consumidor. Los datos de salida generados por una parte de la aplicación se usan como entrada de otra.
c) Modelo equipo: Todos los hilos se comportan como iguales. Cada hilo obtiene y procesa solicitudes de clientes por sí mismo. Se usa para la implementación de hilos especializados dentro de un proceso.



Diseño de hilos

Un sistema que de soporte a hilos debe proveer un conjunto de primitivas o paquete de hilos:

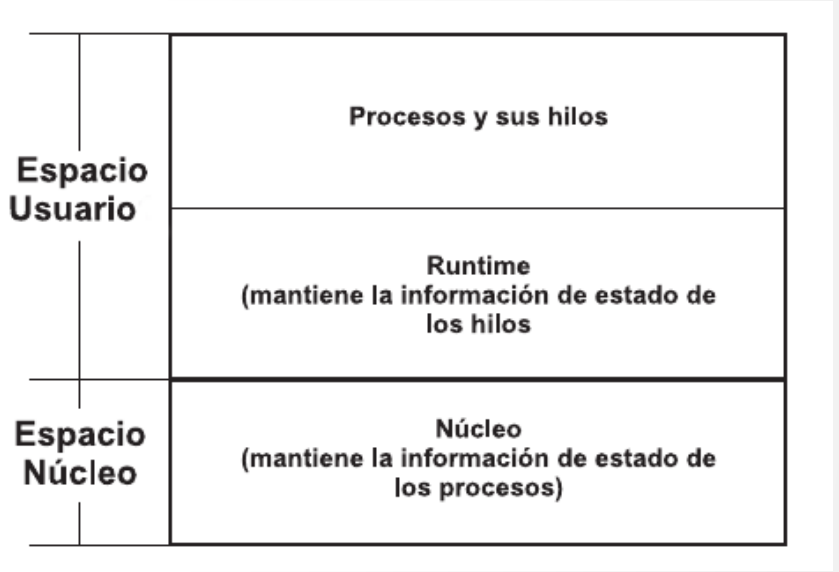
- Creación de los hilos: de forma estática o dinámica.
- Terminación de los hilos: de manera similar a los procesos, cuando finaliza su trabajo o al recibir una señal externa.
- Sincronización: se usan mecanismos para prevenir que hilos accedan de forma simultánea a recursos.
- Planificación:
  - Asignación de prioridades
  - Variar el quantum dinámicamente
  - Planificación forzada
  - Planificación por prioridad
- Manejo de señales: Las señales proveen interrupciones generadas por software y excepciones. Las interrupciones son disrupciones generadas externamente a un hilo (o proceso), mientras que las excepciones son causadas por la ocurrencia de condiciones inusuales durante la ejecución de un hilo.



Implementación

Un paquete de hilos puede ser implementado ya sea en espacio de **usuario** o en el **núcleo**.

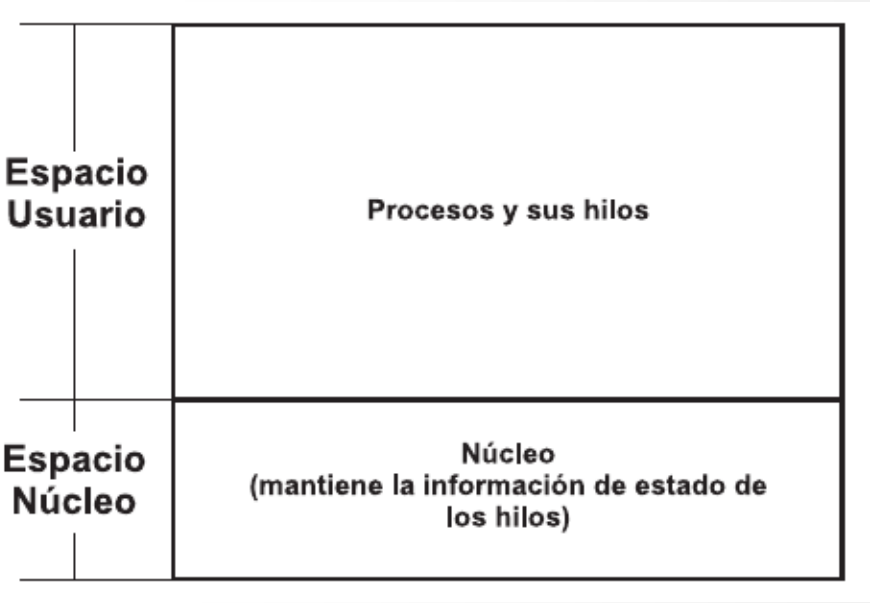
- En **espacio de usuario**: compuesto de un *runtime*. Los hilos corren sobre este y son manejados por él.
- Las llamadas del paquete se implementan como llamada a los procedimientos del sistema *runtime*.
- El planificador en el núcleo asigna quantums a procesos pesados, y el planificador del runtime divide un quantum asignado a un proceso entre los hilos.
- La existencia de hilos es invisible para el núcleo.
- Este enfoque es el usado por el paquete *Lightweight Processes* de SunOS 4.1.



Implementación

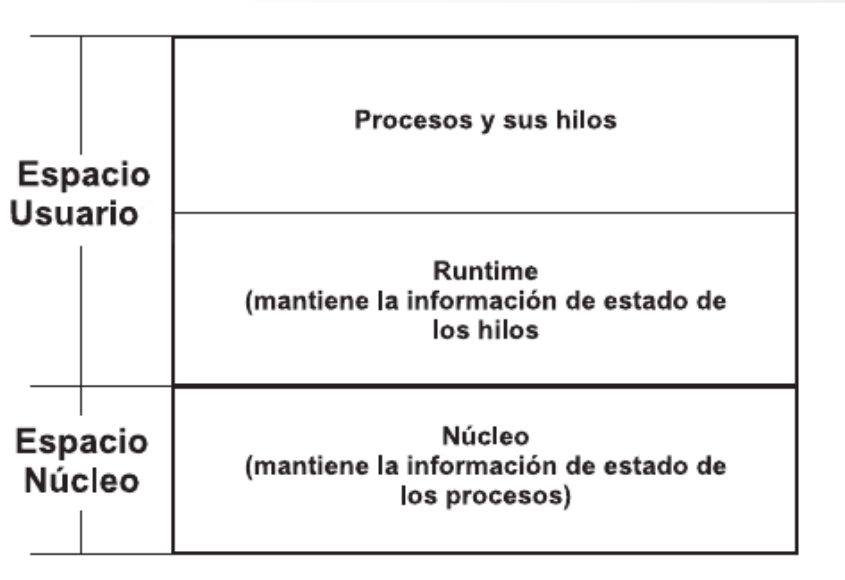
Un paquete de hilos puede ser implementado ya sea en espacio de **usuario** o en el **núcleo**.

- En **espacio de núcleo**: no se usa un sistema *runtime*. Los hilos son manejados por el núcleo.
- La tabla de información de estado de los hilos es mantenida dentro del núcleo.
- Todas las llamadas que deberían bloquear a un hilo se implementan como llamadas a sistema que pisan la trampa al núcleo.
- Cuando un hilo se bloquea, el núcleo selecciona otro hilo para ser ejecutado.
- La existencia de hilos es conocida por el núcleo.



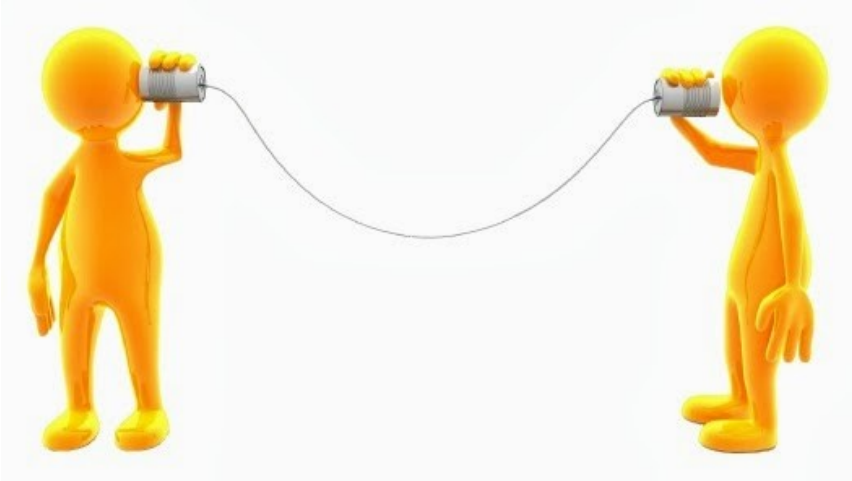
Ventajas y desventajas de los dos enfoques

- La ventaja principal del enfoque del sistema runtime es que estos paquetes pueden implementarse sobre cualquier sistema operativo.
- En el enfoque de nivel de usuario, gracias a la planificación en dos niveles, los usuarios pueden usar sus propios algoritmos para los hilos de un proceso. En el otro se usa un planificador de un único nivel dentro del núcleo.
- La permutación de contexto de un hilo a otro es más rápido en el primero porque lo lleva a cabo el sistema runtime, en el otro es necesaria una trampa al núcleo.
- No es posible una política de planificación por turno circular debido a una falta de interrupciones de reloj.
- Dificultad en llamadas a sistema bloqueantes. A un hilo no se le deberíapermitir hacer llamadas a sistemas en forma directa porque detendría a los demás hilos.



Comunicación

- Normalmente los procesos se comunican con otros.
- Cuando estos procesos se juntan o **concurren** en el mismo espacio y tiempo y acceden a los mismos recursos, deberemos establecer alguna forma de **sincronización**.
- Un proceso es **independiente** si no afecta o es afectado por la ejecución de otros procesos en el sistema.
- Un proceso **cooperativo** es uno que puede afectar o ser afectado por los demás procesos que se ejecutan en el sistema.



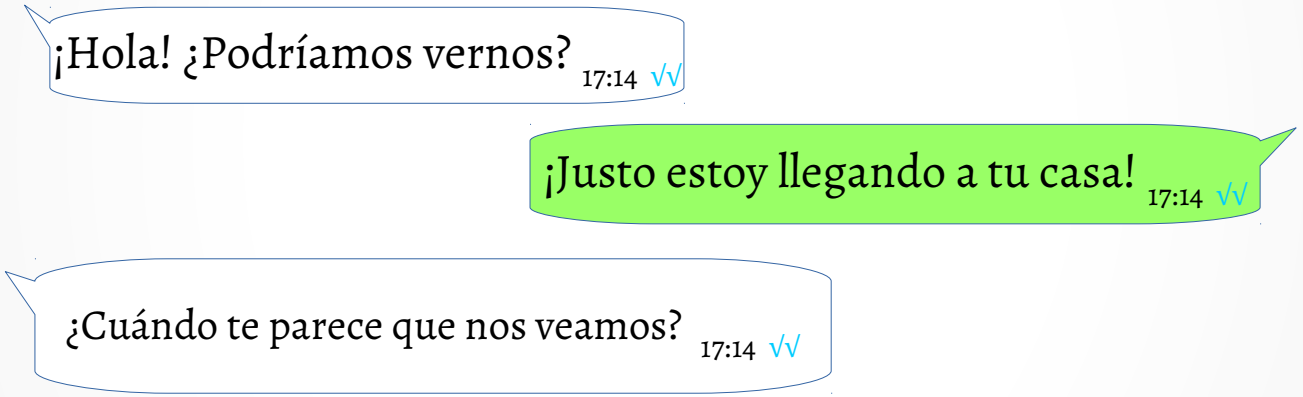
- Dos procesos son concurrentes cuando la primera instrucción de uno de ellos se ejecuta después de la primera instrucción del otro y antes de la última. Hay solapamiento en la ejecución de sus instrucciones.
- Dos eventos son **concurrentes** si mediante la observación del programa no podemos establecer cuál ocurrirá primero.

Entonces: Una característica más acertada de **proceso** es la de una **actividad asíncrona**, susceptible de ser asignada a un procesador (Palma Méndez, 2003).

UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO

Comunicación y sincronización

- Para llevar a cabo estas tareas de colaboración y competencia por los recursos, se hace necesaria la introducción de mecanismos de comunicación y sincronización entre procesos.



Procesos concurrentes

Según la RAE, **concurrencia** es "acaecimiento o concurso de varios sucesos en un mismo tiempo".

- Para llevar a cabo estas tareas de colaboración y competencia por los recursos, se hace necesaria la introducción de mecanismos de comunicación y sincronización entre procesos.

x=x+1;
y=x+2;
	x=1;
	y=2;
	z=3;

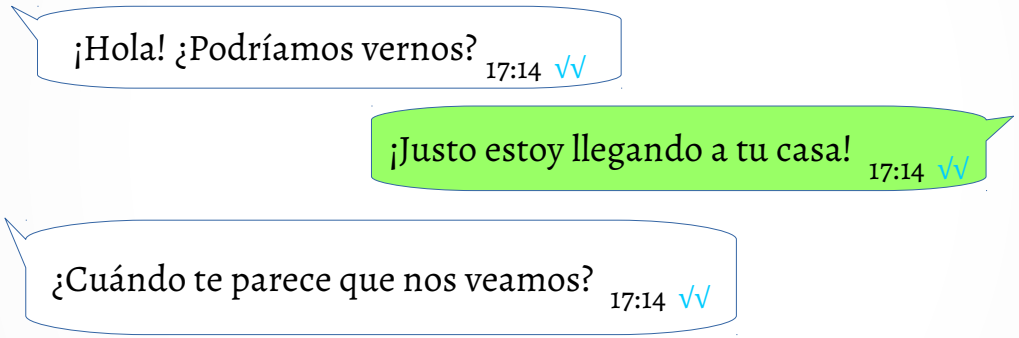
- En el primer caso es claro que la primera sentencia se tiene que ejecutar antes que la segunda.
- En cambio en el segundo caso queda claro que el orden en que se ejecuten no interviene en el resultado final.
- Si dispusiésemos de tres procesadores, podríamos ejecutar cada una de las líneas en uno de ellos, incrementando la velocidad del sistema. Por lo tanto, diremos que:

Dos eventos son **concurrentes** si mediante la observación del programa no podemos establecer cuál ocurrirá primero.

Procesos concurrentes

La ejecución de sistemas concurrentes tiene algunas características que los diferencia de los sistemas secuenciales

- El orden de ejecución de las instrucciones y el indeterminismo. En los programas **secuenciales** hay un **orden total** en la ejecución de las líneas de código. Ante un conjunto de datos de entrada se abe siempre por dónde va a ir el programa.



- En los programas **concurrentes** hay un **orden parcial**, ante el mismo conjunto de datos de entrada no se puede saber cuál va a ser el flujo de ejecución. En cada ejecución del programa el flujo puede ir por distintos sitios.
- Este orden parcial lleva a que los programas concurrentes puedan tener un comportamiento indeterminista, es decir, puedan arrojar diferentes resultados cuando se ejecutan repetidamente sobre el mismo conjunto de datos de entrada.

Paradigmas de comunicación entre procesos

La comunicación entre procesos (comúnmente IPC, del inglés *Inter-Process Communication*) es una función básica de los sistemas operativos. Las dos maneras básicas de abordar este tema son

1) Memoria compartida
2) Pase de mensajes