Sistemas Operativos

Especialización en Sistemas Embebidos

Dr. Mario Marcelo Berón

Univerisdad Nacional de San Luis

14 de octubre de 2017

Sistemas Operativos

Conceptualización

- Máquina Extendida: Proporciona abstracciones que le simplifican las tareas a los programadores (visión top-down).
- Administrador de Recursos: La tarea del sistema operativo es proporcionar una asignación ordenada y controlada de los procesadores, memorias y dispositivos de E/S entre los diversos programas que compiten con estos recursos. El sistema operativo lleva un registro de qué programa está utilizando qué recurso, de otorgar peticiones de recursos, y mediar conflictos en las peticiones de recursos provenientes de programas de usuario.

Sistemas Operativos

Tipos de Sistemas Operativos

- Sistemas Operativos de Mainframe (linux)
- Sistemas Operativos de Servidores (Solaris, FreeBSD, Linux, Windows Server 200x)
- Sistemas Operativos de Multiprocesadores (Windows, Linux)
- Sistemas Operativos de Computadoras Personales (Linux, FreeBSD, Windowas, Mac OS).
- Sistemas Operativos de Computadoras de Bolsillo (Symbiam OS, Palm OS)
- Sistemas Operativos Integrados (QNX, VxWorks)
- Sistemas Operativos de Nodos Sensores (TinyOS)
- Sistemas Operativos de Tiempo Real (eCos)
- Sistemas Operativos de Tarjetas Inteligentes

Sistemas Operativos

Conceptos de Sistemas Operativos

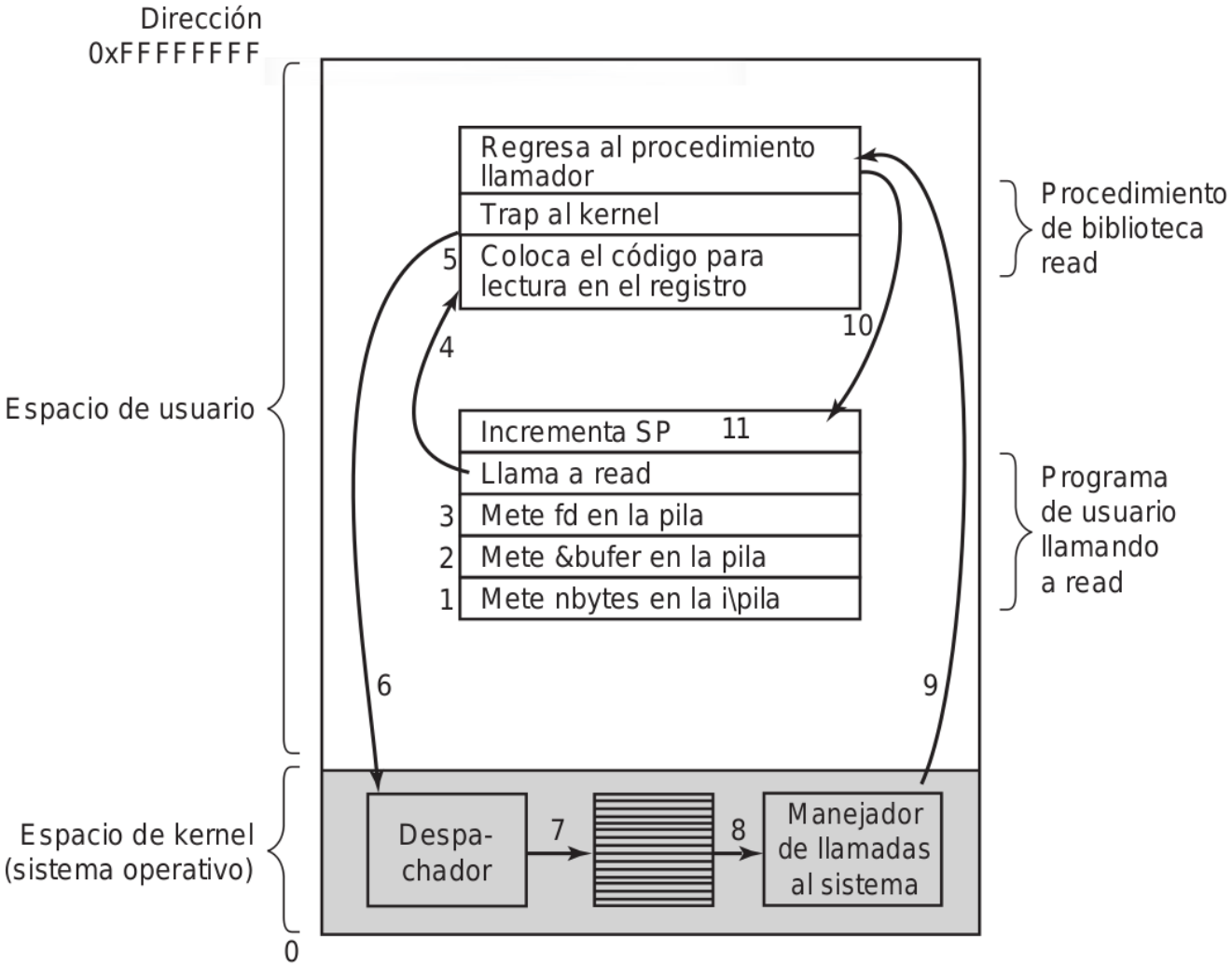
- Proceso
- Espacio de Direcciones
- Archivos
- Entrada-Salida
- Intérprete de Comandos

Sistemas Operativos

Llamadas al Sistema

Interfaz entre las aplicaciones de usuario y el sistema operativo que facilita interactuar con el sistema operativo.

Sistemas Operativos



Sistemas Operativos

Estructura de un Sistema Operativo

- Sistemas Operativos Monolíticos
- Sistemas Operativos de Capas
- Microkernels
- Sistemas Operativos Cliente-Servidor
- Máquinas Virtuales
- Exokernels

Sistemas Operativos

Sistema Operativo Monolítico

El sistema operativo se escribe como una colección de procedimientos enlazados entre sí en un único programa ejecutable.

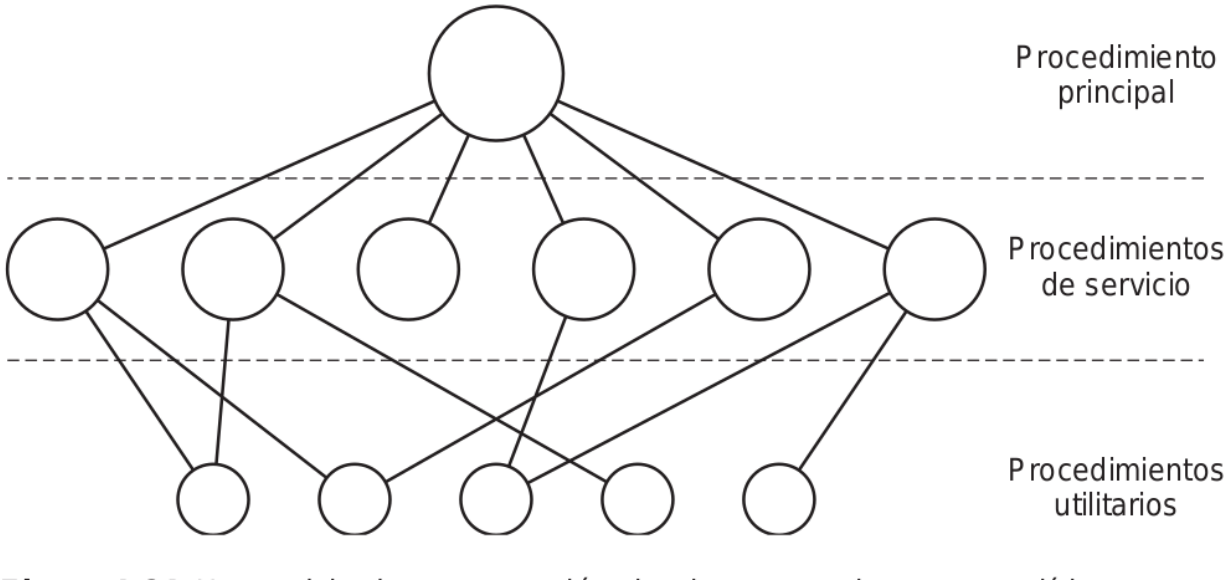
Importante

Cada procedimiento en el sistema tiene libertad de llamar a cualquier otro si éste proporciona un cómputo útil que el primero necesita. Al tener muchos procedimientos este tipo de sistema operativo puede ser difícil de mantener y comprender.

Sistemas Operativos

Sistema Operativo Monolítico

El sistema operativo se escribe como una colección de procedimientos enlazados entre sí en un único programa ejecutable.



Comentarios

En este modelo para cada llamada al sistema hay un procedimiento de servicio que se encarga de la llamada y la ejecuta.

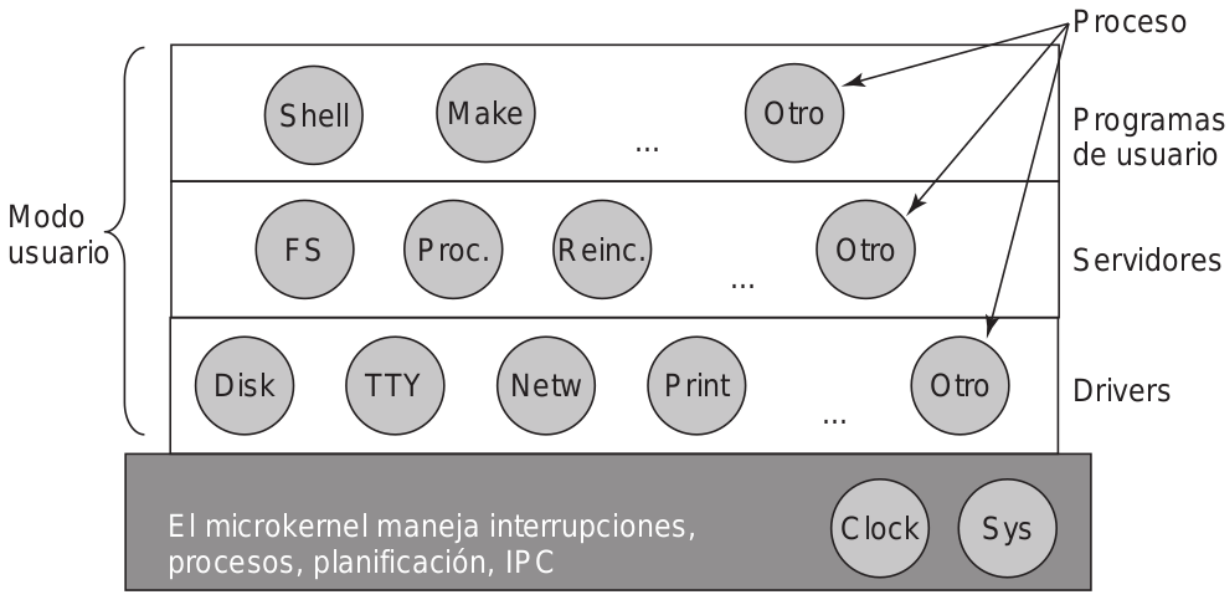
Sistema Operativo en Capas

El sistema operativo se construye como una jerarquía de capas, cada capa se construye encima de la que tiene abajo.

Capa	Función
5	El operador
4	Programas de usuario
3	Administración de la entrada/salida
2	Comunicación operador-proceso
1	Administración de memoria y tambor
0	Asignación del procesador y multiprogramación

Microkernels

El núcleo del sistema operativo es pequeño para evitar que el sistema se paralice de inmediato. La idea básica es dividir el sistema operativo en módulos pequeños y bien definidos, solo uno de los cuales se ejecuta en modo núcleo: el microkernel y el resto se ejecuta como procesos ordinarios sin poder.



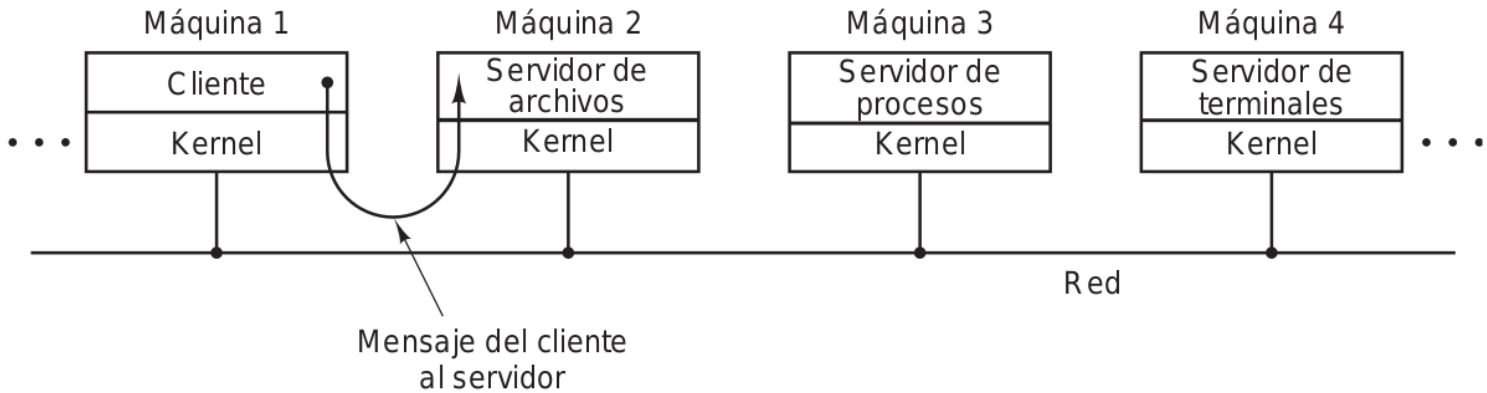
Modelo Cliente-Servidor

Una variación del microkernel consiste en diferenciar dos clases de procesos: los servidores, los cuales proporcionan una clase de servicio, y los clientes que utilizan esos servicios.

- La comunicación entre el servidor y los clientes se lleva a cabo mediante el pasaje de mensajes.
- Para obtener un servicio el cliente construye un mensaje que indica que necesita dicho servicio y lo envías al servicio apropiado.
- El servicio hace el trabajo y envía la respuesta.

Modelo Cliente-Servidor

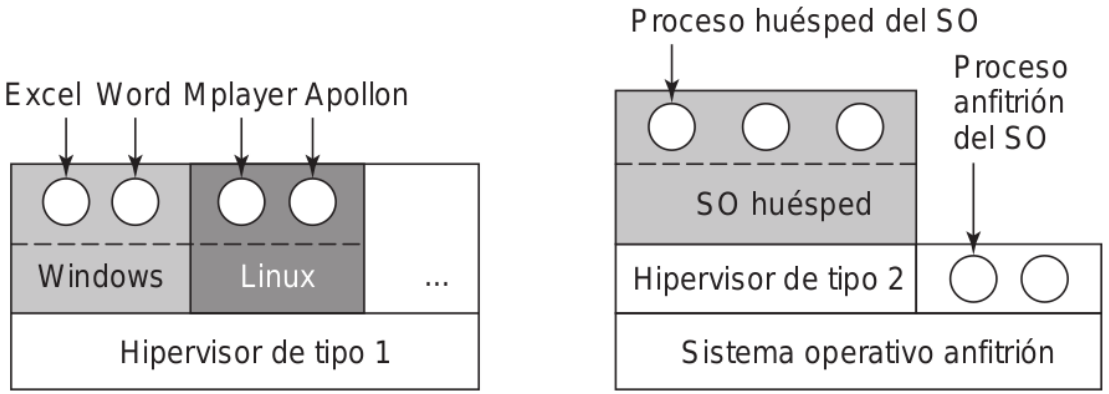
Una generalización obvia es hacer que los clientes y servidores se ejecuten en distintas computadoras conectadas mediante una red de área local o amplia



Sistemas Operativos

Maquinas Virtuales

Son copias exactas del hardware incluyendo modo kernel/usuario,la E/S, las interrupciones y todo lo que tiene una máquina real.



Comentario

Uno de los usos de la virtualización es para los usuarios finales que desean ejecutar dos o más sistemas operativos al mismo tiempo, esto se realiza comunmente con Linux y Windows, debido a que algunos de sus paquetes se ejecutan en el primero y otros en el segundo.

Sistemas Operativos

Exokernels

En vez de clonar la máquina actual, como se hace con las máquinas virtuales, otra estrategia es particionarla; en otras palabras, a cada usuario se le proporciona un subconjunto de los recursos. Así, una máquina virtual podría obtener los bloques de disco del 0 al 1023, la siguiente podría obtener los bloques de disco del 1024 al 2047 y así sucesivamente. En la capa inferior, que se ejecuta en el modo kernel, hay un programa llamado exokernel. Su trabajo es asignar recursos a las máquinas virtuales y después comprobar los intentos de utilizarlos, para asegurar que ninguna máquina trate de usar los recursos de otra

Ventaja

La ventaja del esquema del exokernel es que ahorra una capa de asignación.

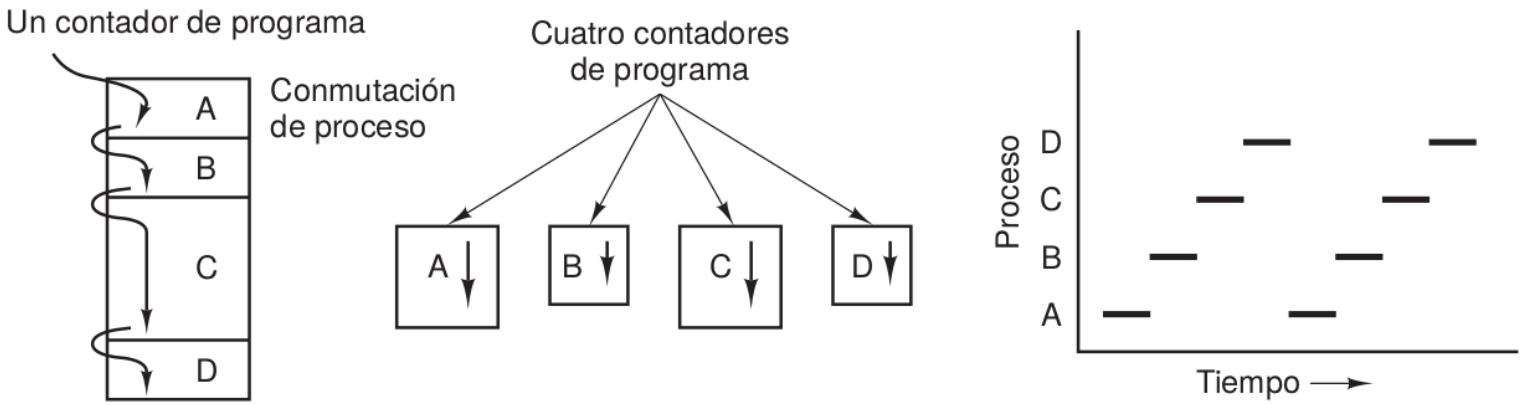
Procesos e Hilos

Procesos

Programa en ejecución incluyendo el contador de programa, los registros, las variables, etc.

Modelo de Procesos

La CPU conmuta de proceso en proceso, esta conmutación rápida se conoce como *Multiprogramación*.



Procesos e Hilos

Creación de un Proceso

- El arranque de un sistema.
- La ejecución, desde un proceso, de una llamada al sistema para creación de procesos.
- Una petición de usuario para crear un proceso.
- El inicio de un trabajo por lotes.

Procesos e Hilos

Finalización de Procesos

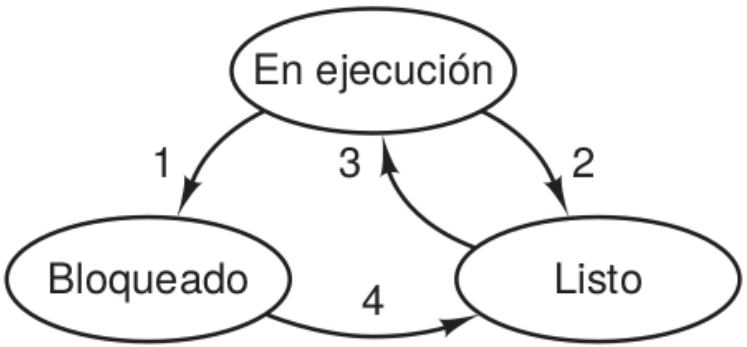
- Salida normal voluntaria (voluntaria).
- Salida por error(voluntaria).
- Error falta (involuntaria).
- Eliminado por otro proceso (involuntaria).

Jerarquía de Procesos

En algunos sistemas, cuando un proceso crea otro, el proceso padre y el proceso hijo continúan asociados en ciertas formas. El proceso hijo puede crear por sí mismo más procesos, formando una jerarquía de procesos.

Procesos e Hilos

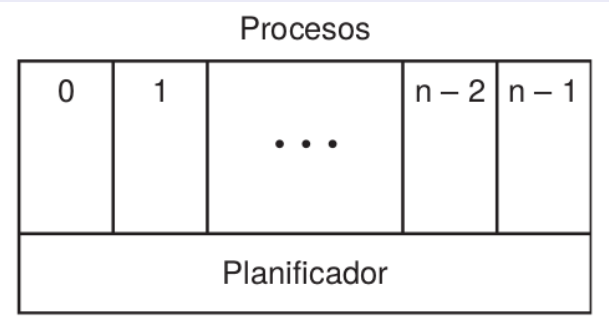
Estados de un Proceso



1. El proceso se bloquea para recibir entrada
2. El planificador selecciona otro proceso
3. El planificador selecciona este proceso
4. La entrada ya está disponible

Implementación de Procesos

Se debe mantener una tabla de procesos. Esta tabla mantiene información importante del proceso, como por ejemplo: el contador de programas, el puntero a la pila, asignación de memoria, estado de los archivos abiertos, etc.



Procesos e Hilos

Implementación de Procesos

Algunos de los datos que se almacenan en la tabla de procesos se muestran en la figura de abajo.

Administración de procesos	**Administración de memoria**	**Administración de archivos**
Registros	Apuntador a la información del segmento de texto	Directorio raíz
Contador del programa	Apuntador a la información del segmento de datos	Directorio de trabajo
Palabra de estado del programa	Apuntador a la información del segmento de pila	Descripciones de archivos
Apuntador de la pila		ID de usuario
Estado del proceso		ID de grupo
Prioridad		
Parámetros de planificación		
ID del proceso		
Proceso padre		
Grupo de procesos		
Señales		
Tiempo de inicio del proceso		
Tiempo utilizado de la CPU		
Tiempo de la CPU utilizado por el hijo		
Hora de la siguiente alarma		

Procesos e Hilos

Implementación de Procesos

1. El hardware mete el contador del programa a la pila, etc.
2. El hardware carga el nuevo contador de programa del vector de interrupciones.
3. Procedimiento en lenguaje ensamblador guarda los registros.
4. Procedimiento en lenguaje ensamblador establece la nueva pila.
5. El servicio de interrupciones de C se ejecuta (por lo general lee y guarda la entrada en el búfer).
6. El planificador decide qué proceso se va a ejecutar a continuación.
7. Procedimiento en C regresa al código de ensamblador.
8. Procedimiento en lenguaje ensamblador inicia el nuevo proceso actual.

Procesos e Hilos

Hilos

Con frecuencia hay situaciones en las que es conveniente tener varios hilos de control en el mismo espacio de direcciones que se ejecuta en cuasi-paralelo, como si fueran procesos (casi) separados (excepto por el espacio de direcciones compartido).

Procesos e Hilos

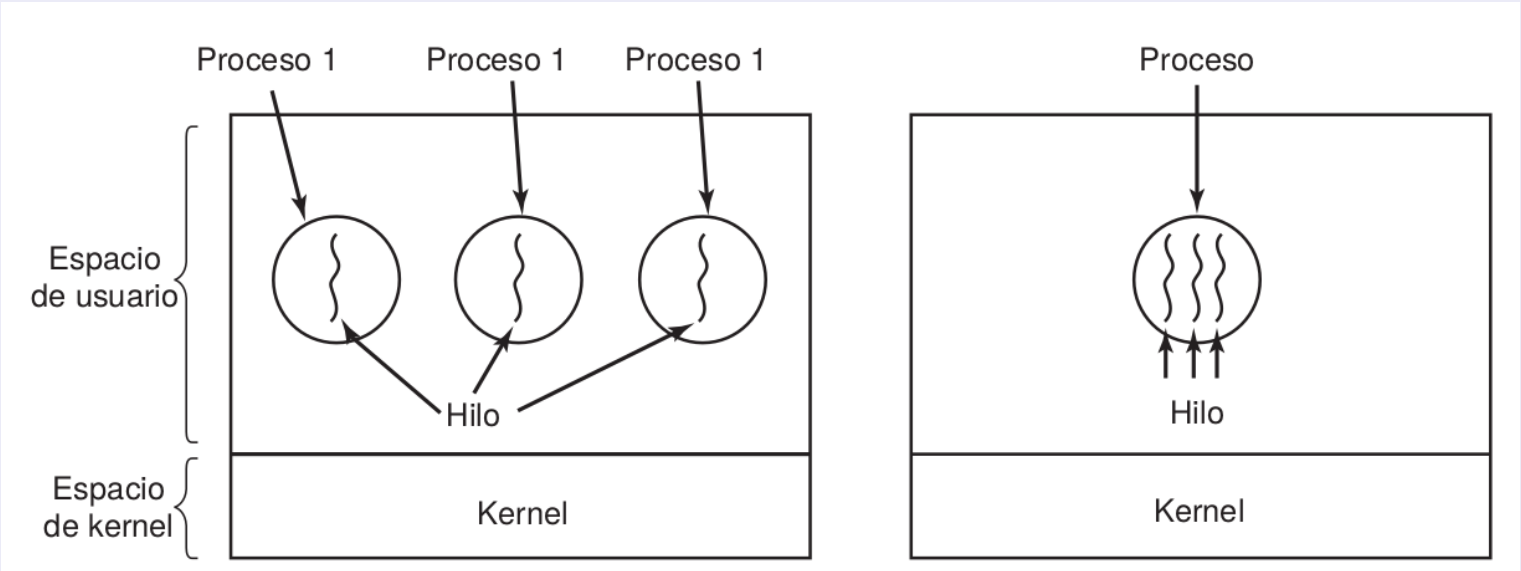
Hilos

¿Por qué hilos?

- El modelo de programación se simplifica se piensa solo en procesos paralelos.
- La creación de un hilo es más rápida que la creación de un proceso.
- Los hilos no producen un aumento en el rendimiento cuando todos ellos están ligados a la CPU, pero cuando hay una cantidad considerable de cálculos y operaciones de E/S, al tener hilos estas actividades se pueden traslapar, con lo cual se agiliza la velocidad de la aplicación.
- Son útiles en sistemas con varias CPU en donde es posible el paralelismo.

Procesos e Hilos

Modelo Clásico de Hilos



Procesos e Hilos

Modelo Clásico de Hilos

Elementos por proceso	**Elementos por hilo**
Espacio de direcciones	Contador de programa
Variables globales	Registros
Archivos abiertos	Pila
Procesos hijos	Estado
Alarmas pendientes	
Señales y manejadores de señales	
Información contable	

Procesos e Hilos

Problemas

- ¿qué ocurre si un hilo en el padre se bloqueó en una llamada read , por ejemplo, del teclado? ¿Hay ahora dos hilos bloqueados en el teclado, uno en el padre y otro en el hijo? Cuando se escriba una línea, ¿obtendrán ambos hilos una copia de ella? ¿Será sólo para el padre? ¿O sólo para el hijo? El mismo problema existe con las conexiones abiertas de red.
- ¿Qué ocurre si un hilo cierra un archivo mientras otro aún está leyendo datos de él? Suponga que un hilo detecta que hay muy poca memoria y empieza a asignar más. A mitad del proceso, ocurre una conmutación de hilos y el nuevo hilo también detecta que hay muy poca memoria y también empieza a asignar más memoria. Es muy probable que se asigne memoria dos veces.

Procesos e Hilos

Comunicación Interprocesos

En esta temática se abordan tres problemas:

1. Como un proceso puede pasar información a otro proceso.
2. Evitar que dos procesos se interpongan entre si.
3. Obtener la secuencia apropiada cuando hay dependencias presentes.

Procesos e Hilos

Condiciones de Carrera

La condición de carrera (race condition) ocurre cuando dos o más procesos acceden un recurso compartido sin control, de manera que el resultado combinado de este acceso depende del orden de llegada.

Condiciones de Carrera

Depurar programas que contienen condiciones de carrera no es nada divertido. Los resultados de la mayoría de las ejecuciones de prueba están bien, pero en algún momento poco frecuente ocurrirá algo extraño e inexplicable.

Procesos e Hilos

Exclusión Mutua

Asegurar que si un proceso está utilizando una variable o archivo compartido los demás procesos se excluirán de hacer lo mismo.

Región Crítica

Parte del programa que accede al recurso compartido.

Región Crítica

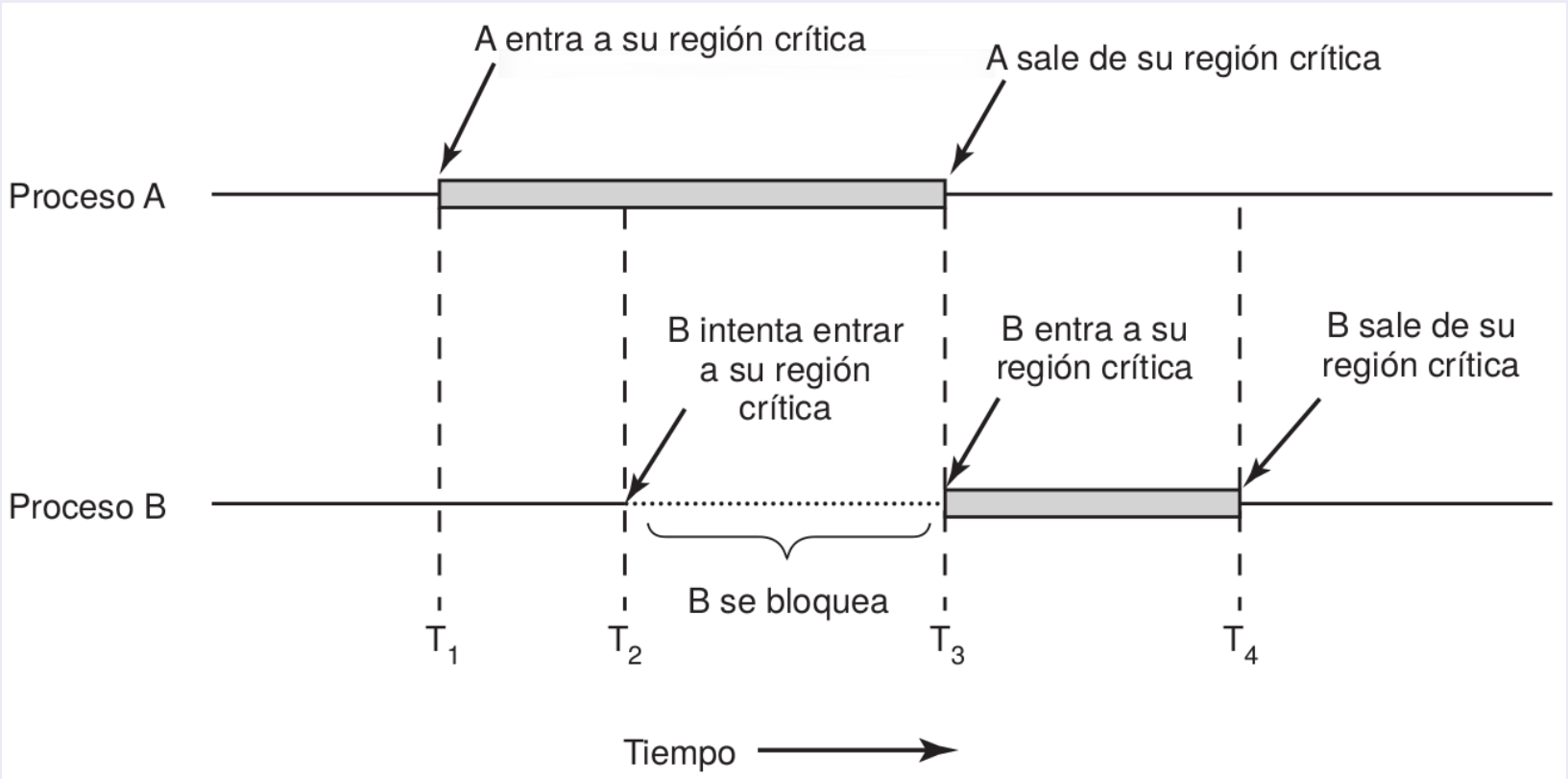
Si se pueden ordenar las cosas de manera que dos procesos nunca estén en su región crítica se pueden evitar las carreras.

Evitar Condiciones de Carrera

Si se pueden ordenar las cosas de manera que dos procesos nunca estuvieran en su región crítica se pueden evitar las carreras.
Además se tiene que cumplir:

- No puede haber dos procesos de manera simultánea en una región crítica.
- No pueden hacerse suposiciones acerca de las velocidades o el número de CPUs.
- Ningún proceso que se ejecute fuera de su región crítica puede bloquear otros procesos.
- Ningún proceso tiene que esperar para siempre para entrar a su región crítica.

Evitar Condiciones de Carrera



Procesos e Hilos

Exclusión Mutua con Espera Ocupada

- Deshabilitar Interrupciones
- Variables Candado
- Alternancia Estricta
- Solución de Peterson

Exclusión Mutua: Alternancia Estricta

```
while (TRUE) {
    while (turno != 0)        /* ciclo */ ;
    region_critica();
    turno = 1;
    region_nocritica();
}
```

(a)

```
while (TRUE) {
    while (turno != 1)        /* ciclo */ ;
    region_critica();
    turno = 0;
    region_nocritica();
}
```

(b)

Procesos e Hilos

Exclusión Mutua: Solución de Peterson

```
#define FALSE 0
#define TRUE 1
#define N   2                                   /* número de procesos */

int turno;                                      /* ¿de quién es el turno? */
int interesado[N];                              /* al principio todos los valores son 0 (FALSE) */

void entrar_region(int proceso);                /* el proceso es 0 o 1 */
{
    int otro;                                   /* número del otro proceso */

    otro = 1 – proceso;                         /* el opuesto del proceso */
    interesado[proceso] = TRUE;                 /* muestra que está interesado */
    turno = proceso;                            /* establece la bandera */
    while (turno == proceso && interesado[otro] == TRUE)  /* instrucción nula */;
}

void salir_region(int proceso)                  /* proceso: quién está saliendo */
{
    interesado[proceso] = FALSE;                /* indica que salió de la región crítica */
}
```

Exclusión Mutua sin Espera Ocupada: Semáforos

- Tienen dos operaciones asociadas down y up.
- down: comprueba si el valor es mayor que 0. Si este es el caso disminuye el valor y continua ejecutando. Si el valor es 0 el proceso se bloquea.
- up: incrementa el valor asociado al semáforo. Si uno o más procesos estaban inactivos en ese semáforo, sin poder completar una operación down anterior, el sistema selecciona uno de ellos (al azar) y permite que complete su operación down . Así, después de una operación up en un semáforo que contenga procesos dormidos, el semáforo seguirá en 0 pero habrá un proceso menos dormido en él.

Exclusión Mutua sin Espera Ocupada: Semáforos

Veamos algunos ejemplos concretos