Introducción a los Sistemas Operativos

Introducción – III

Profesores:
Lía Molinari
Juan Pablo Pérez
Macia Nicolás

Facultad de Informática
UNIVERSIDAD NACIONAL DE LA PLATA

I.S.O.

- ☑ Versión: Agosto 2016
- ☑ Palabras Claves: Sistema Operativo, Hardware, System Call, Interacción, Modos de Ejecución, Protección, Kernel

Los temas vistos en estas diapositivas han sido mayormente extraídos del libro de Andrew S. Tanenbaum (Sistemas Operativos Modernos)

Problemas que un SO debe evitar

- ☑ Que un proceso se apropie de la CPU
- ☑ Ejecutar instrucciones de E/S ilegales
- ☑ Intentar acceder a una posición ilegal de memoria
  - Proteger los espacios de direcciones

Para ello, el SO entre otras debe:

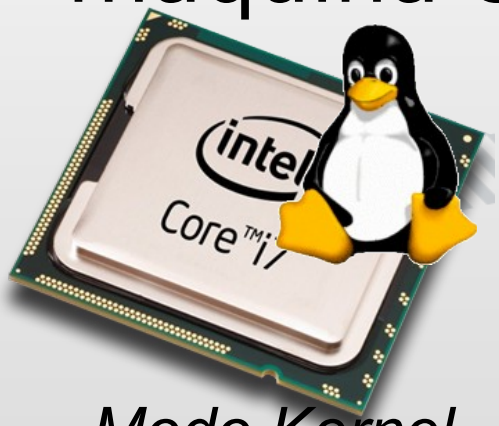
- ☑ Gestionar el uso de la CPU
- ☑ Detectar intentos de ejecución de instrucciones de E/S ilegales
- ☑ Detectar accesos ilegales a memoria
- ☑ Proteger el vector de interrupciones
  - Así como las RAI (Rutinas de atención de interrupciones)

Modos de ejecución - Usuario y Supervisor

- El bit de modo indica el modo actual
- Las instrucciones privilegiadas deben ejecutarse en modo supervisor (necesitan acceder a estructuras del kernel, o ejecutar código que no es del proceso)
- Cuando está en modo usuario, el proceso puede acceder sólo a su espacio de direcciones, es decir a las direcciones “propias”.

Modos de ejecución (cont)

- El código del SO se ejecuta en modo supervisor
- Por lo tanto: en el modo usuario sólo un subconjunto de instrucciones de máquina es permitido.



Modo Kernel

Modo Usuario

Tener en cuenta que...

- ☑ Cuando se arranque el sistema, arranca con el bit en modo supervisor.
- ☑ Cada vez que comienza a ejecutarse un proceso de usuario, este bit se DEBE PONER en modo usuario.
  - Mediante una Instrucción especial.
- ☑ Cuando hay un trap o una interrupción, el bit de modo se pone en modo Kernel.
  - Única forma de pasar a Modo Kernel
  - No es el proceso de usuario quien hace el cambio explicitamente.

Cómo actúa...

- Cuando el proceso de usuario intenta por sí mismo ejecutar instrucciones que pueden causar problemas (las llamadas instrucciones privilegiadas) esto el HW lo detecta como una operación ilegal y produce un trap al SO.

En Windows...

- En WIN2000 el modo núcleo ejecuta los servicios ejecutivos. El modo usuario ejecuta los procesos de usuario.
- Cuando un programa se bloquea en modo usuario, a lo sumo se escribe un suceso en el registro de sucesos. Si el bloqueo se produce estando en modo supervisor se genera la BSOD (pantalla azul de la muerte).

Modos de Ejecución

- Procesador Intel 8088 no tenía modo dual de operación ni protección por hardware.
- En MsDos las aplicaciones pueden acceder directamente a las funciones básicas de E/S para escribir directamente en pantalla o en disco.

Resumiendo...

- Modo kernel:
  - Modo privilegiado
  - Manejo estricto de pautas de confiabilidad/seguridad
  - Manejo de:
    - CPU, memoria, operaciones de entrada/salida
    - Administración multiprocesador, diagnosticos, testing
    - Estructura del Filesystem
    - Comunicaciones: interfaz de red
- Modo user:
  - Más flexible
  - Funciones de Mantenimiento más simples, debugging
    - Compiler, assembler, interpreter, linker/loader
    - File system management, telecommunication, network management
    - Editors, spreadsheets, user applications

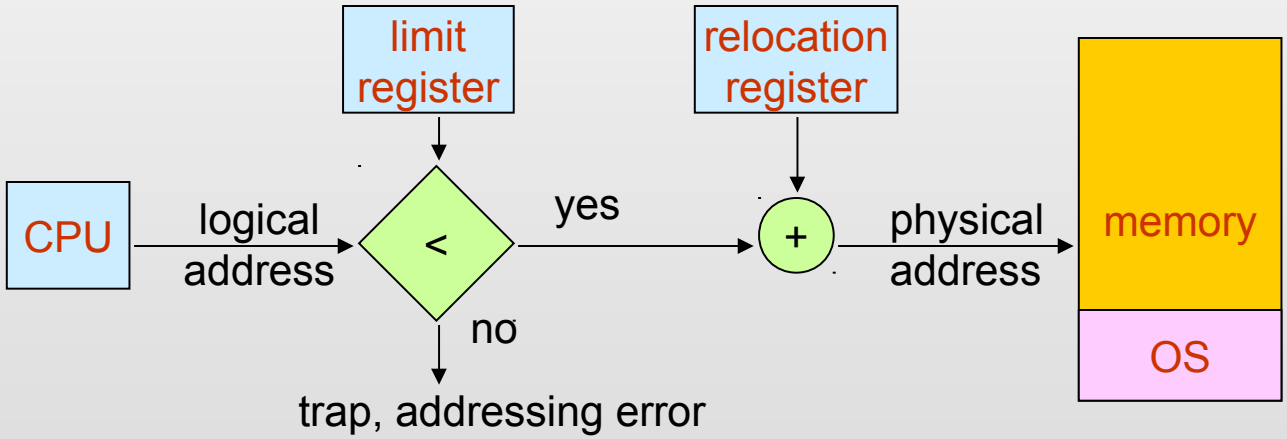
Protección de la memoria

- ☑ Delimitar el espacio de direcciones del proceso
- ☑ Poner limites a las direcciones que puede utilizar un proceso
  - Por ejemplo: Uso de un registro base y un registro límite
  - El SO carga estos registros por medio de instrucciones privilegiadas. Esta acción sólo puede realizarse en modo Kernel

Protección de la memoria (cont)

La memoria principal aloja al SO y a los procesos de usuario

- ✓ El SO debe protegerse de ser accedido por procesos de usuario
- ✓ El debe proteger el espacio de direcciones de un proceso del acceso de otros procesos



Protección de la E/S

- Las instrucciones de E/S se definen como privilegiadas.
- Deben ejecutarse en Modo Kernel
  - Se deberían realizar a través del sistema operativo

Protección de la CPU

- Uso de interrupción por clock para evitar que un proceso se apropie de la CPU
- Se implementa normalmente a través de un clock y un contador.
- El SO le da valor al contador que se decrementa con cada tick de reloj y al llegar a cero se produce la expulsión del proceso

Protección de la CPU (cont.)

- Las instrucciones que modifican el funcionamiento del reloj son privilegiadas.
- Se le asigna al contador el valor que se quiere que se ejecute un proceso.
- Se la usa también para el cálculo de la hora actual, basándose en cantidad de interrupciones ocurridas cada tanto tiempo y desde una determinada fecha y hora.

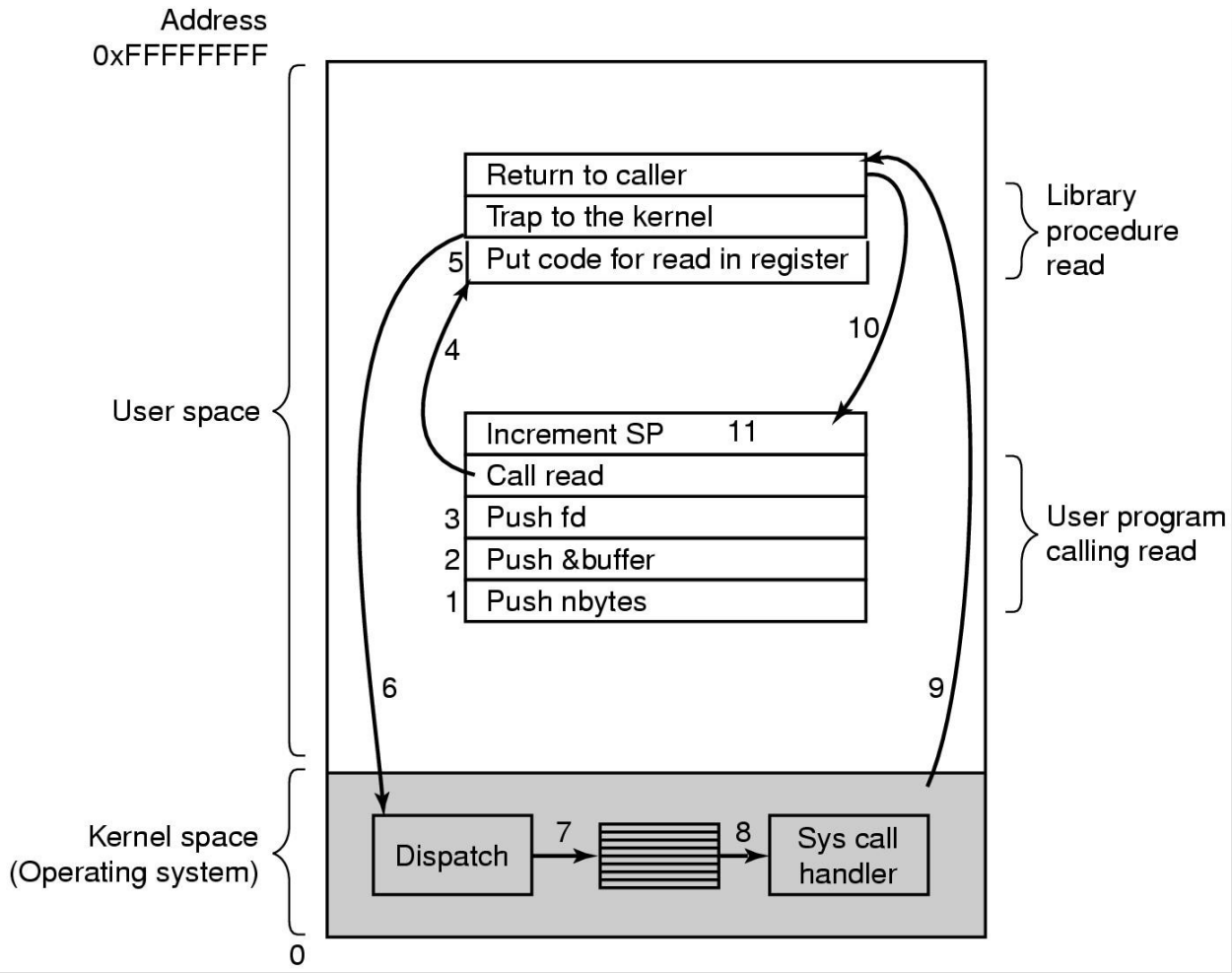
System Calls

- Es la forma en que los programas de usuario acceden a los servicios del SO.
- Los parámetros asociados a las llamadas pueden pasarse de varias maneras: por registros, bloques o tablas en memoria ó pilas.

count=read(file, buffer, nbytes);

- Se ejecutan en modo supervisor

System calls (cont.)



System Calls - Categorias

- Categorías de system calls:
  - Control de Procesos
  - Manejo de archivos
  - Manejo de dispositivos
  - Mantenimiento de información del sistema
  - Comunicaciones

System calls - Categorias (cont.)

Process management

Call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

System calls - Categorias (cont.)

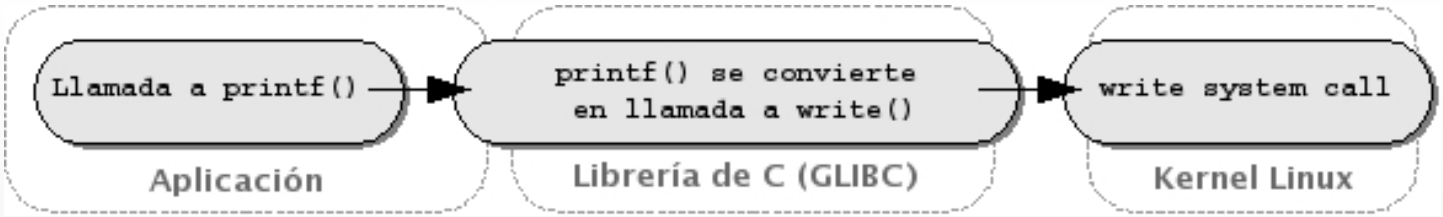
Directory and file system management

Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

Miscellaneous

Call	Description
s = chdir(dirname)	Change the working directory
s = chmod(name, mode)	Change a file's protection bits
s = kill(pid, signal)	Send a signal to a process
seconds = time(&seconds)	Get the elapsed time since Jan. 1, 1970

Ejemplo - System Call Linux

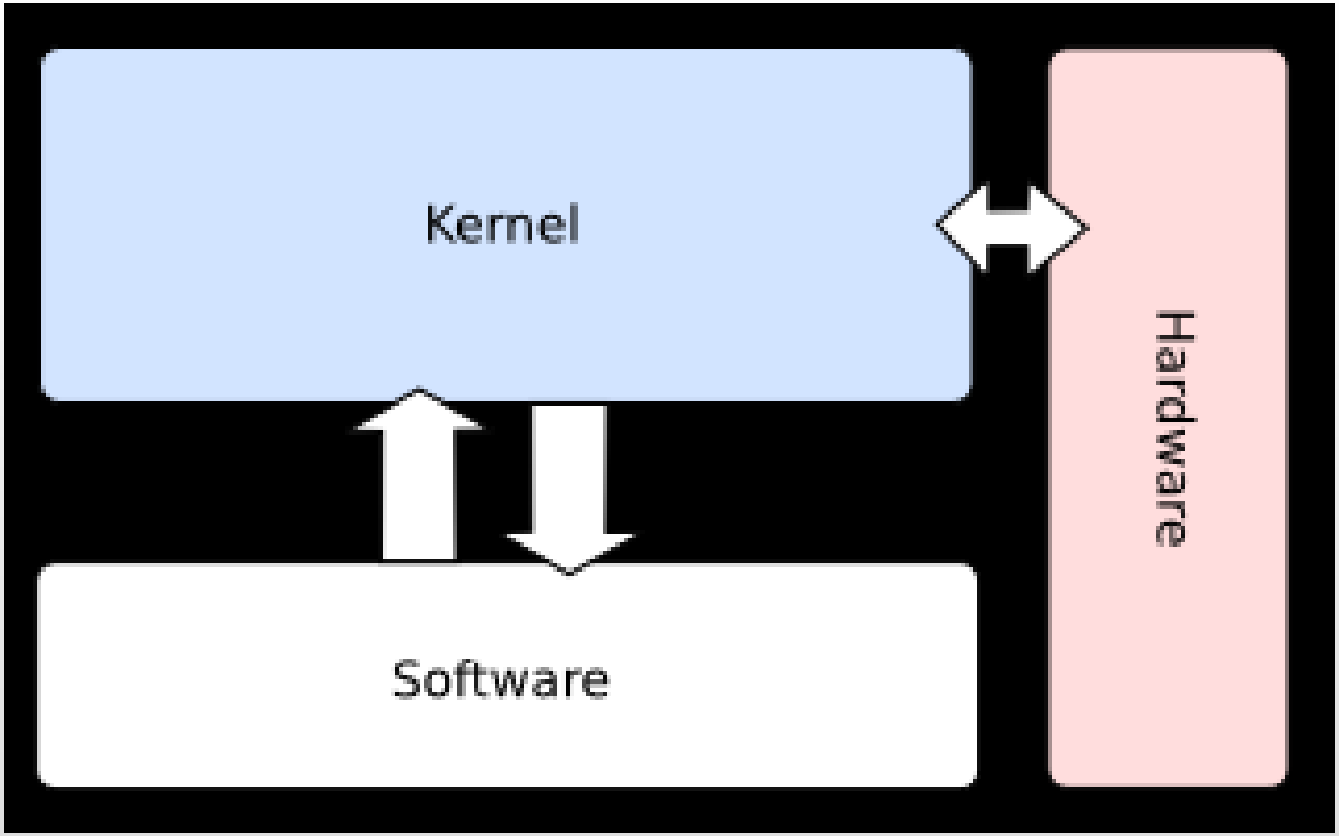


- Se emite una interrupción para invocar al Kernel
  - int $0x80
- Se llama al Interruption Handler
  - System Call Handler
  - El proceso indica con un número la System Call que desea invocar (Syscall Number)

Systems Calls - Ejemplos

UNIX	Win32	Description
fork	CreateProcess	Create a new process
waitpid	WaitForSingleObject	Can wait for a process to exit
execve	(none)	CreateProcess = fork + execve
exit	ExitProcess	Terminate execution
open	CreateFile	Create a file or open an existing file
close	CloseHandle	Close a file
read	ReadFile	Read data from a file
write	WriteFile	Write data to a file
lseek	SetFilePointer	Move the file pointer
stat	GetFileAttributesEx	Get various file attributes
mkdir	CreateDirectory	Create a new directory
rmdir	RemoveDirectory	Remove an empty directory
link	(none)	Win32 does not support links
unlink	DeleteFile	Destroy an existing file
mount	(none)	Win32 does not support mount
umount	(none)	Win32 does not support mount
chdir	SetCurrentDirectory	Change the current working directory
chmod	(none)	Win32 does not support security (although NT does)
kill	(none)	Win32 does not support signals
time	GetLocalTime	Get the current time

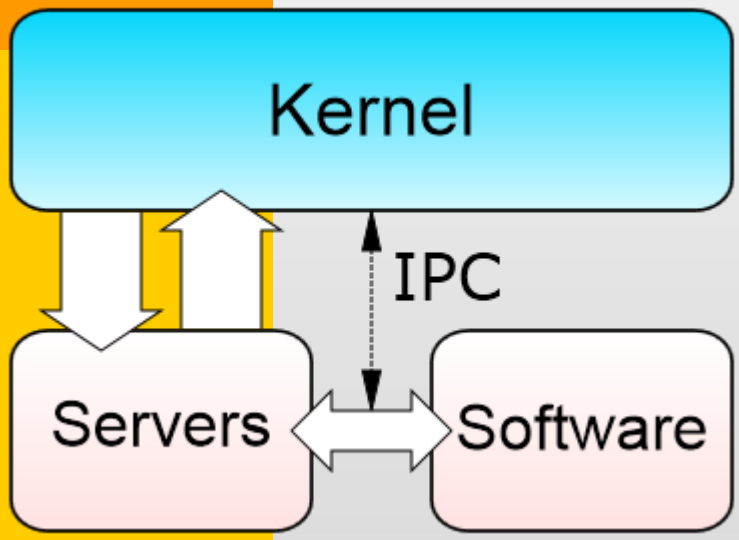
Tipos de kernel - Monolítico



Tipos de kernel - Microkernel

Proceso cliente … Manejador dispositivos Serv. Archivos Serv. procesos Mem. virtual

microkernel



Arquitectura de Windows (simplificada)