

Introducción a los Sistemas Operativos

Introducción – III

Profesores:

Lía Molinari

Juan Pablo Pérez

Macia Nicolás



- ✓ Versión: Marzo 2013
- ✓ Palabras Claves: Sistema Operativo, Hardware, System Call, Interacción, Modos de Ejecución, Protección, Kernel

Algunas diapositivas han sido extraídas de las ofrecidas para docentes desde el libro de Stallings (Sistemas Operativos) y el de Silberschatz (Operating Systems Concepts)



Objetivos de la clase

- ✓ Rol del HW en las tareas del SO
- ✓ Cómo interactúan las aplicaciones con el SO?



Recordando... Servicios del SO

- ✓ Controlar la ejecución de procesos permitiendo su creación, terminación o suspensión y comunicación
- ✓ Planificar el uso de la CPU a través del scheduling de procesos
- ✓ Asignar memoria principal para el proceso en ejecución
- ✓ Asignar memoria secundaria para uso eficiente y recuperación de datos del usuario
- ✓ Permitir el acceso controlado desde los procesos a los dispositivos



Interaccion usuario - SO

- ✓ GUI (Graphical User Interface), basado en gráficos o íconos.
- ✓ Shell o Intérprete de comandos (CLI) basado en texto
- ✓ Ejemplo de uso del shell

```
$ date
```

```
$ date >file
```

```
$ sort <file1 >file2
```

```
$ cat file1 file2 file3 > /dev/lp1
```

```
$ make all >log 2>&1 &
```



Relación del SO con el HW

- ✓ Muchos errores son detectados x el HW

Hay que proteger vector de interrupciones y las rutinas de atención de interrupciones (RAI), para que no sean adulterados

- ✓ Proteger los espacios de direcciones de los procesos



Problemas que un SO debe evitar

- ✓ Producir instrucciones de E/S ilegales
- ✓ Intentar acceder a una posición ilegal de memoria
- ✓ Que un proceso se apropie de la CPU



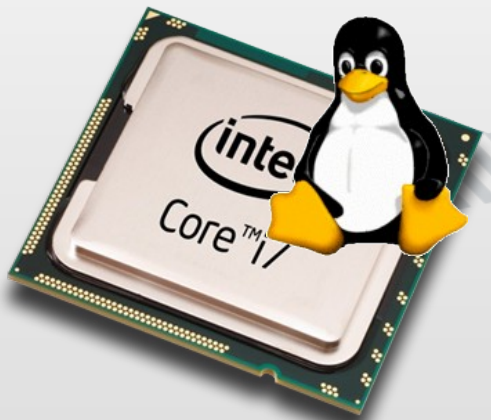
Modos de ejecución - Usuario y Supervisor

- ✓ 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 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.
- ✓ Cuando comienzan a ejecutarse un proceso de usuario, este bit se pone en modo usuario.
- ✓ Cuando hay un trap o una interrupción, el bit de modo se pone en modo supervisor.
- ✓ No es el proceso de usuario quien hace el cambio... Lo hace el HW al detectar una trampa o una interrupción...



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 se bloquea estando en modo supervisor se produce la BSOD , la pantalla azul de la muerte, y el servidor se detiene.



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 a la pantalla o al disco.



Resumiendo...

☑ Modo kernel:

- ✓ Modo privilegiado
- ✓ Manejo estricto de pautas de confiabilidad/seguridad
- ✓ Manejo de:
 - ♦ CPU, memoria, Input/Output
 - ♦ Administración multiprocesador, diagnosticos, testing
 - ♦ Partes del filesystem y la interfase 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

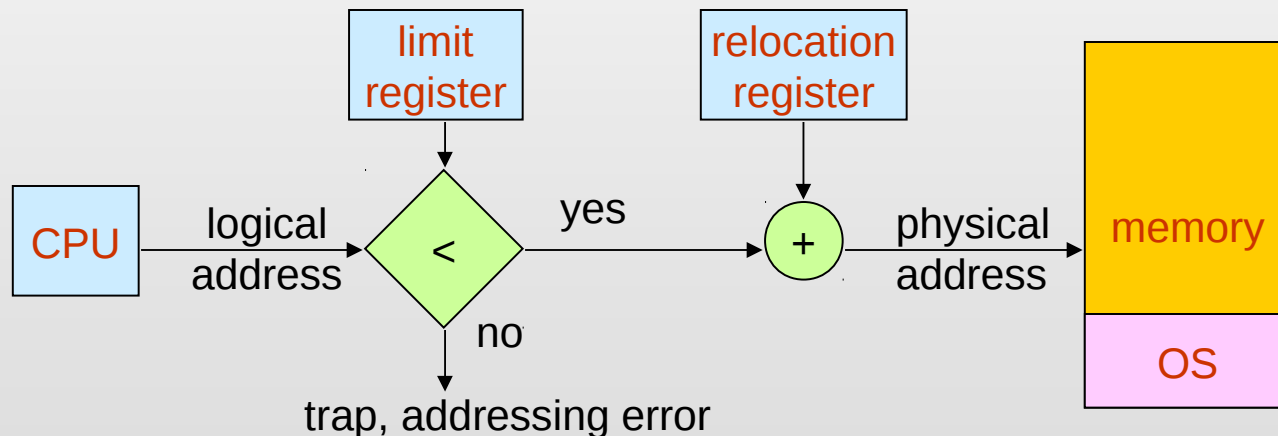
- ✓ Delimitar el espacio de direcciones del proceso
- ✓ Uso de registro base y registro límite
- ✓ El SO puede cargar valores en estos registros a través de instrucciones privilegiadas. Esta acción sólo puede realizarse en modo monitor.



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 hacerse 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 cronómetro 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.



Conceptos importantes: System Calls

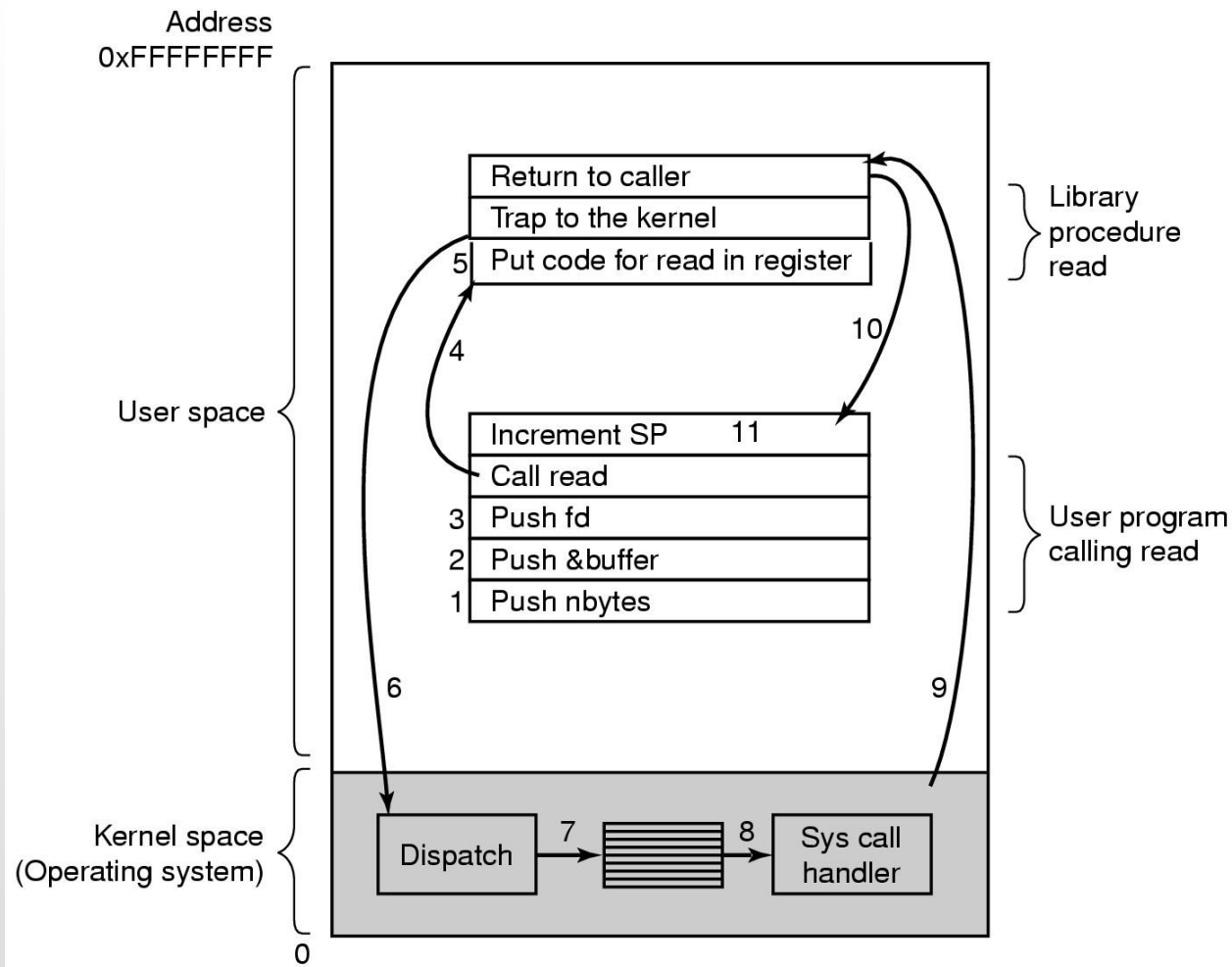
- ✓ Es la forma en que los programas de usuario acceden a los servicios del SO.
- ✓ Los parámetros asociados a las llamadas pueden pasarse de varias maneras: por registros, bloques o tablas en memoria ó pilas.

count=read(file, buffer, nbytes);

- ✓ Se ejecutan en modo supervisor



System calls (cont.)



System Calls - Categorías

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



System calls - Categorías (cont.)

Process management

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

File management

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



System calls - Categorías (cont.)

Directory and file system management

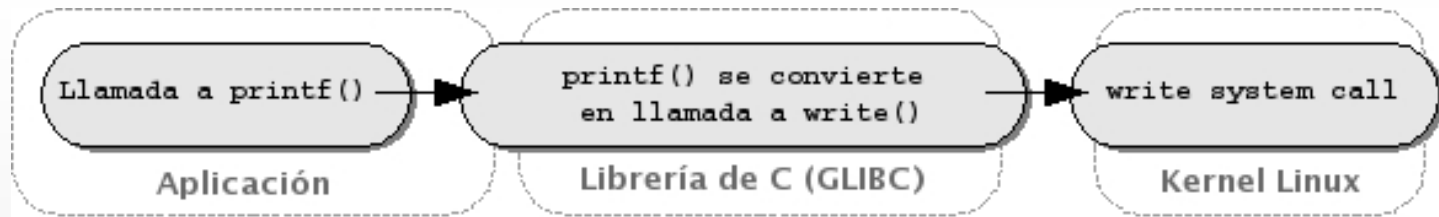
Call	Description
<code>s = mkdir(name, mode)</code>	Create a new directory
<code>s = rmdir(name)</code>	Remove an empty directory
<code>s = link(name1, name2)</code>	Create a new entry, name2, pointing to name1
<code>s = unlink(name)</code>	Remove a directory entry
<code>s = mount(special, name, flag)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

Miscellaneous

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



Ejemplo - System Call Linux 2.6



- ✓ 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



Ejemplo - System Call Linux 2.6 (cont.)

- ✓ Cada System Call tiene un número asignado dentro del kernel
 - ✓ Nunca debe ser alterado ni reutilizado en caso de desaparecer una SysCall
- ✓ `sys_call_table`
 - ✓ `<kernel_code>/arch/i386/kernel/syscall_table.S`

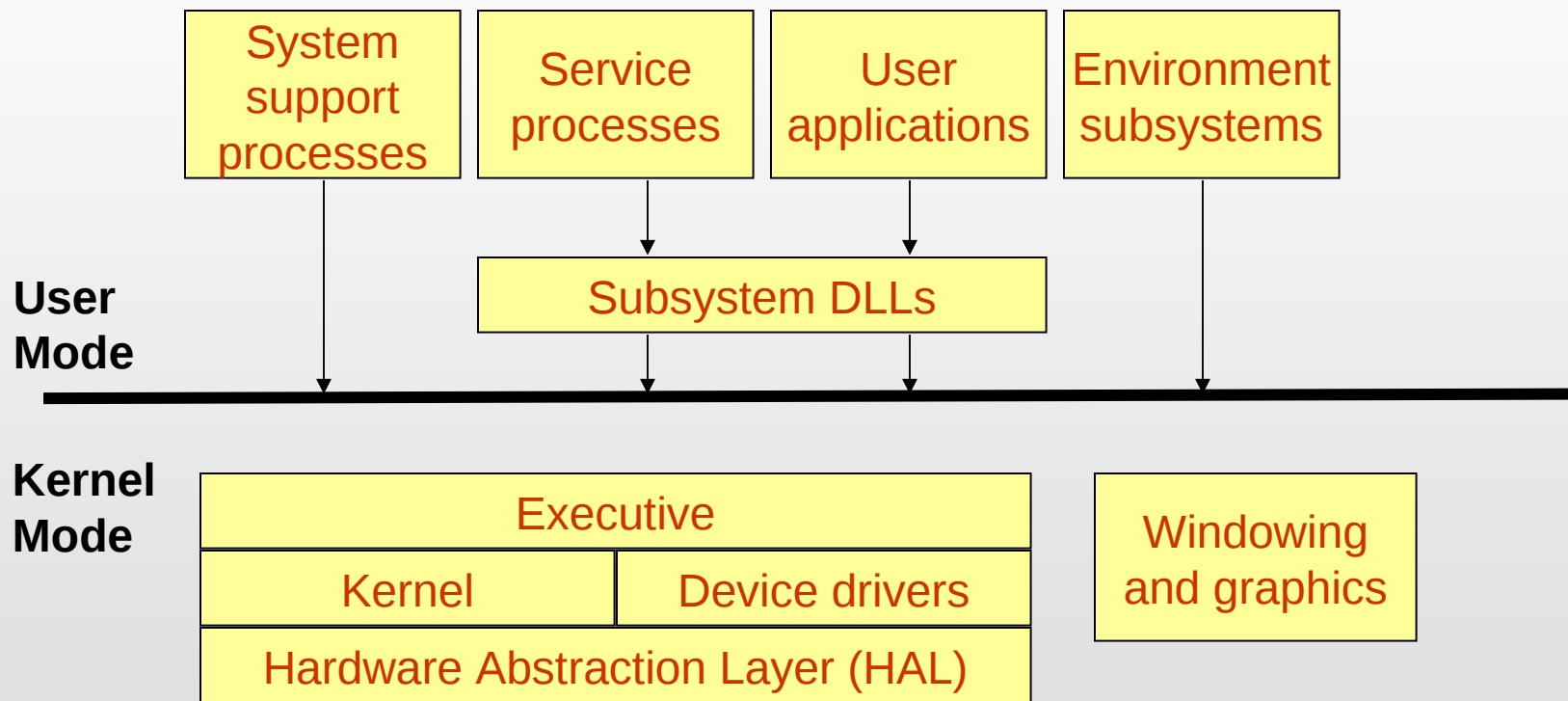


API de Windows y system calls

- ☑ Windows API functions:
 - ✓ Subrutinas invocables, Documentadas.
 - ✓ *CreateProcess*, *CreateFile*, *GetMessage*
- ☑ Windows system services:
 - ✓ Funciones No Documentadas, invocables desde el espacio de Usuario.
 - ✓ *NtCreateProcess* is used by Windows *CreateProcess* and POSIX *fork()* as an internal service
- ☑ Windows internal routines:
 - ✓ Subrutinas dentro del Windows Executive, Kernel, or HAL
 - ✓ Solo son invocables en Kernel Mode. (device driver, NT OS components)
 - ✓ *ExAllocatePool* (Alocar memoria en Windows system heap, utilizada por los drivers)



Arquitectura de Windows (simplificada)

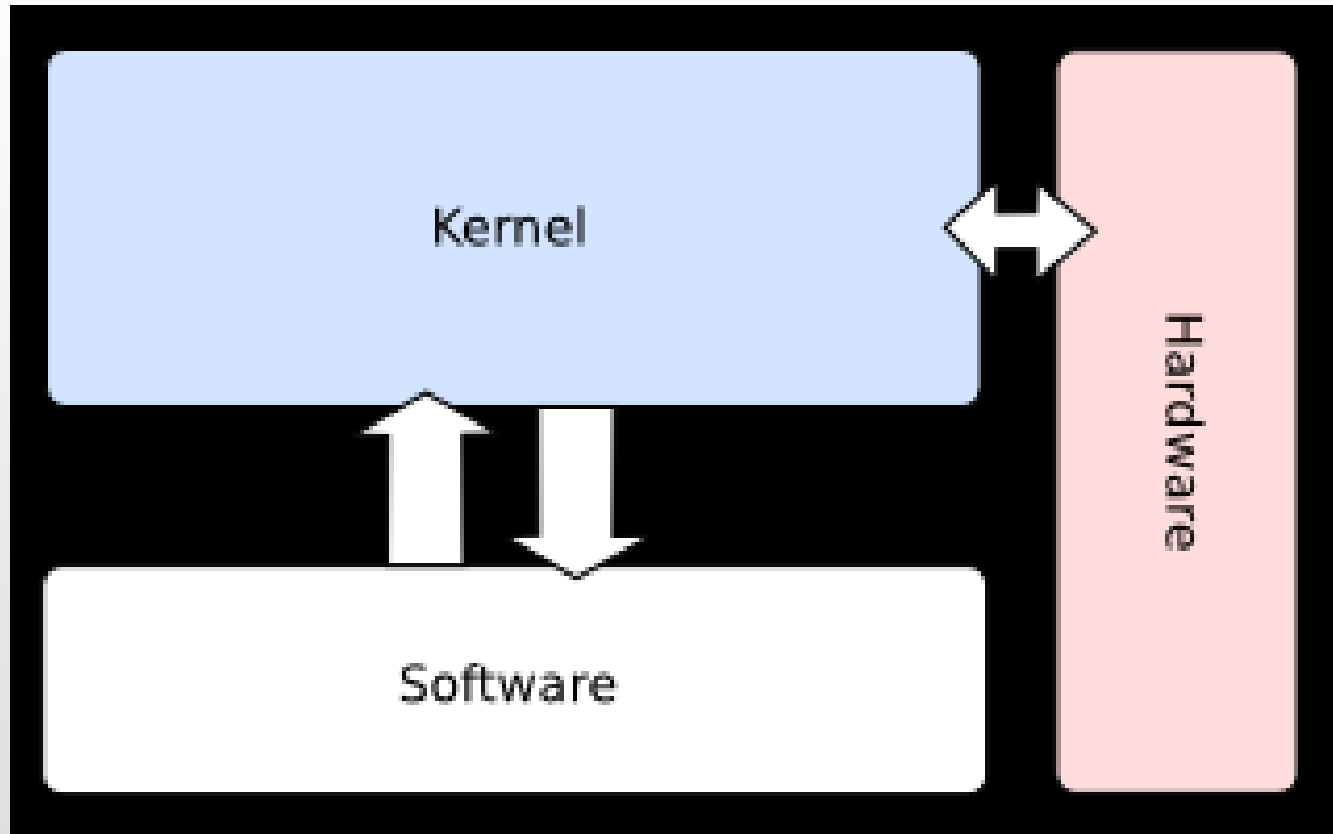


Systems Calls - Ejemplos

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

