.UBAfiuba
FACULTAD DE INGENIERÍA

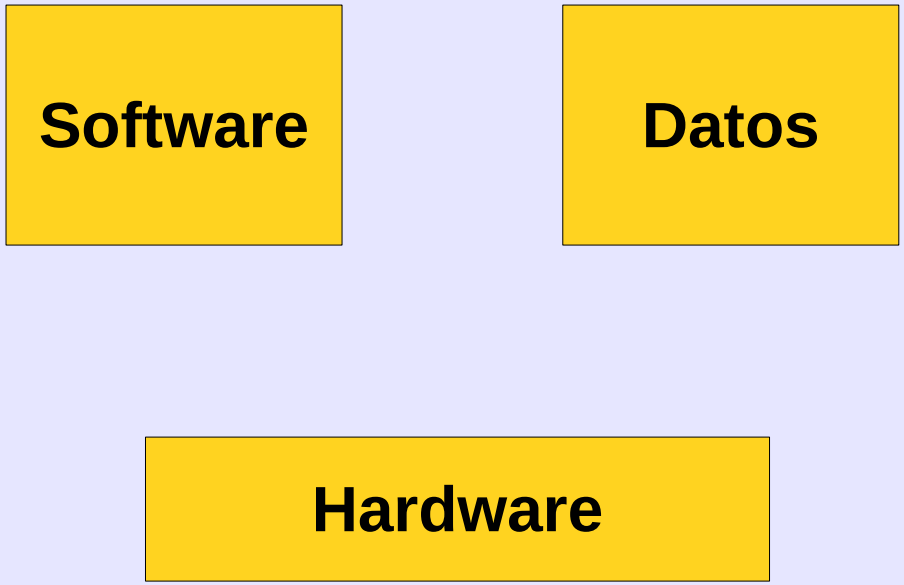
150 ING
el país celebra su ingeniería
1870-2020

75-08 Sistemas Operativos
Lic. Ing. Osvaldo Clúa
Lic. Adrián Muccio

Mecanismos Básicos de un Sistema Operativo

Sistema de Computación

Los Sistemas implementan el concepto de “máquina extendida”

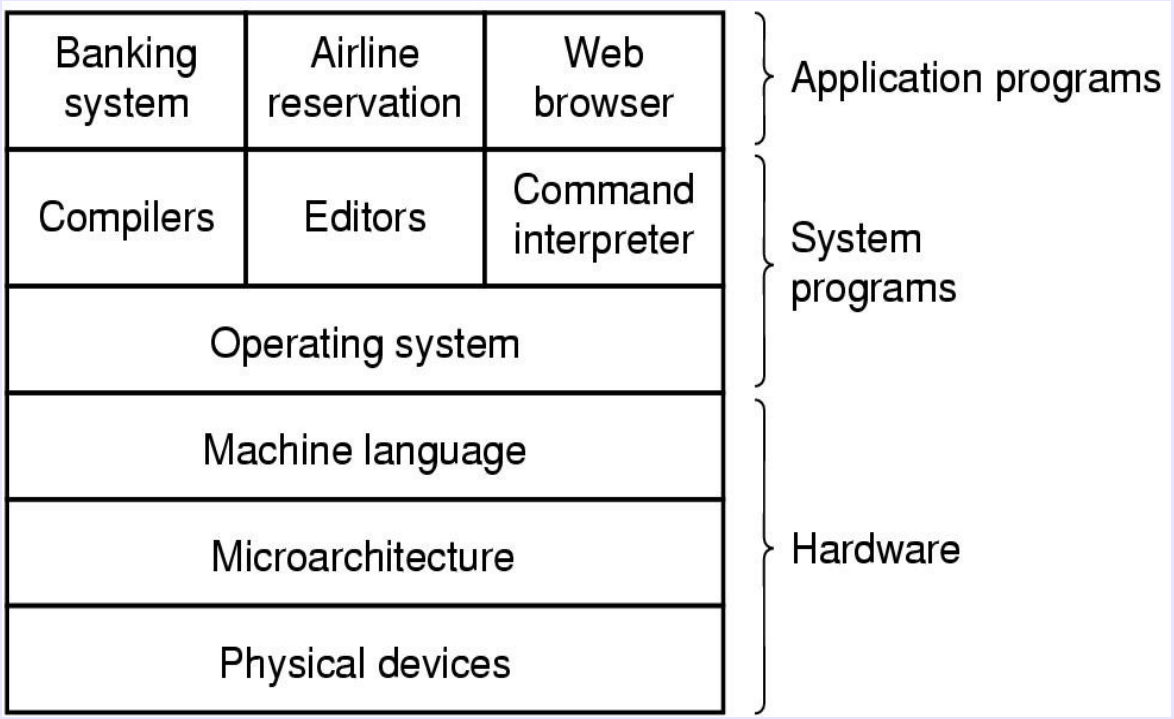


El Hardware provee los recursos al sistema operativo: CPU, memoria y dispositivos I/O

¿Qué pasa cuando se ejecuta un Programa?

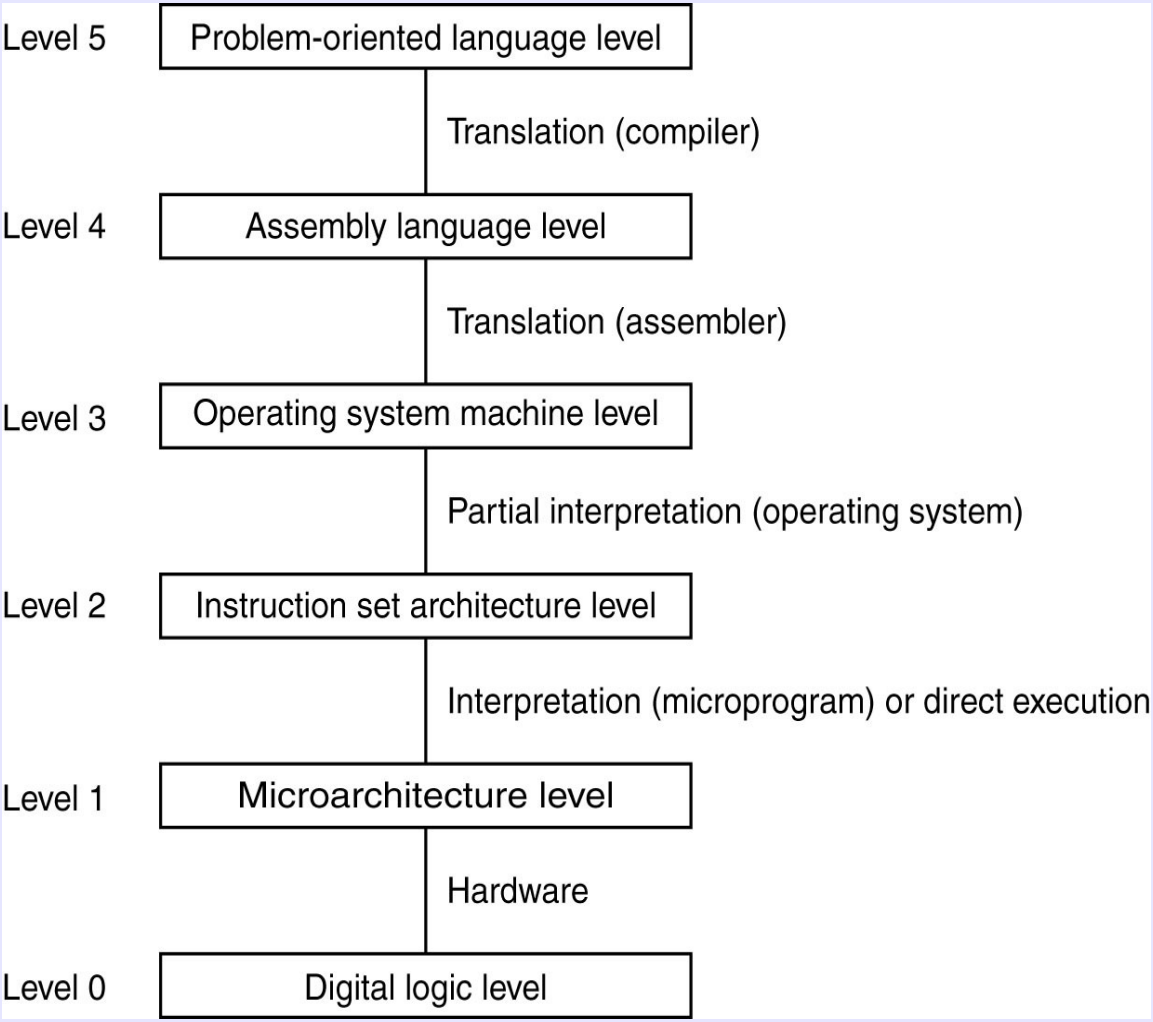
- Se ejecutan instrucciones: Quizá más de una a la vez
- El procesado busca una instrucción (fetch) en la memoria
- La interpreta (decode)
- La ejecuta: Suma o evalúa condición o "salta" a otra instrucción, etc

Concepto de máquina extendida

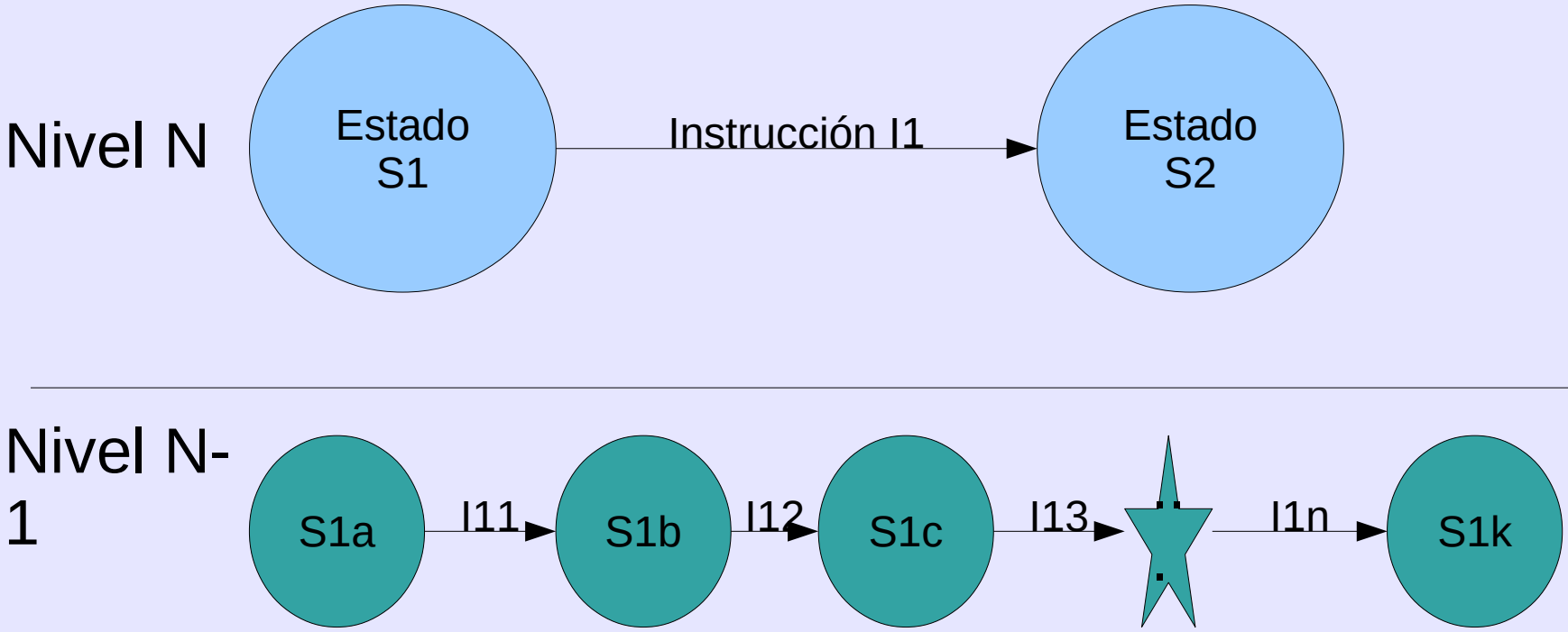


Cada nivel *interpreta* al nivel superior

Niveles


Level 5
Problem-oriented language level
Translation (compiler)
Level 4
Assembly language level
Translation (assembler)
Level 3
Operating system machine level
Partial interpretation (operating system)
Level 2
Instruction set architecture level
Interpretation (microprogram) or direct execution
Level 1
Microarchitecture level
Hardware
Level 0
Digital logic level

Interpretación



El estado de una máquina virtual solo está definido entre instrucción e instrucción.

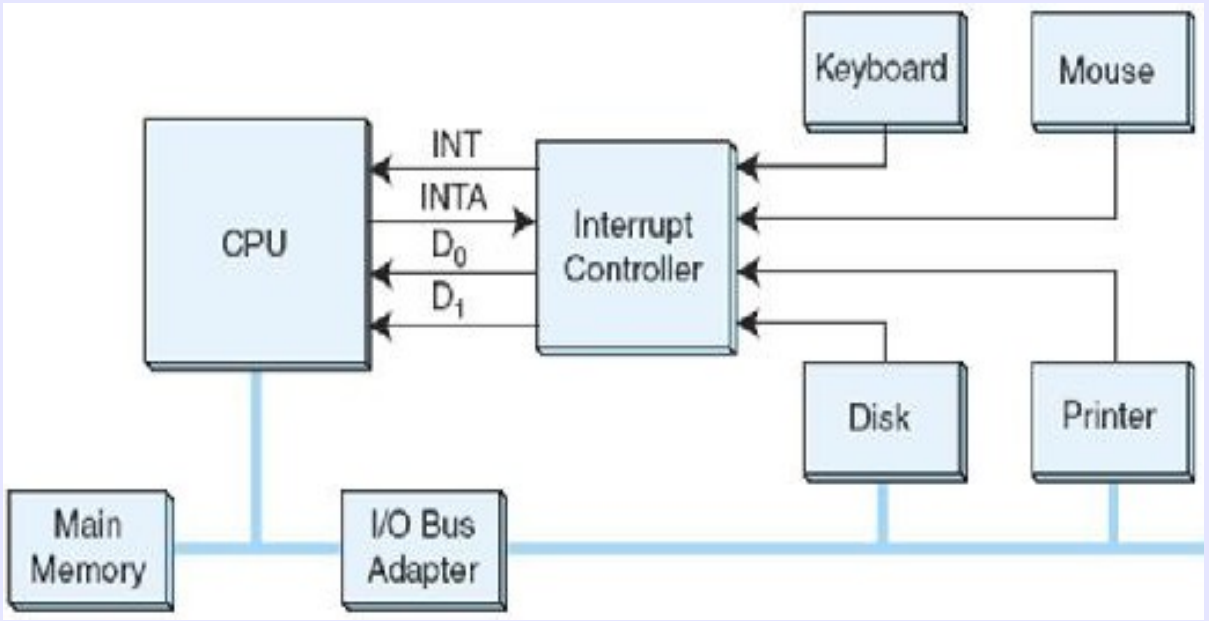
Modos de la CPU (1)

- Son restricciones a las instrucciones que pueden ejecutarse.
- Se los conoce también como niveles o anillos de privilegio.
  - El modo con más privilegios (menos restricciones) se conoce como *ring-0, kernel, master-mode, supervisor, privileged.*
  - El resto son modos de usuario o *user-mode.*

Modos de la CPU (2)

- Se pasa de un modo usuario a modo supervisor por medio de una *Interrupción*.
  - Sincrónica o Software trap.
  - Asincrónica (I/O, Timer, External)
- El retorno a un modo usuario está a cargo del programa.

Interrupciones



- Primer Nivel de atención de interrupciones
  - salvar el contexto (registros, código de condición, dirección de retorno)

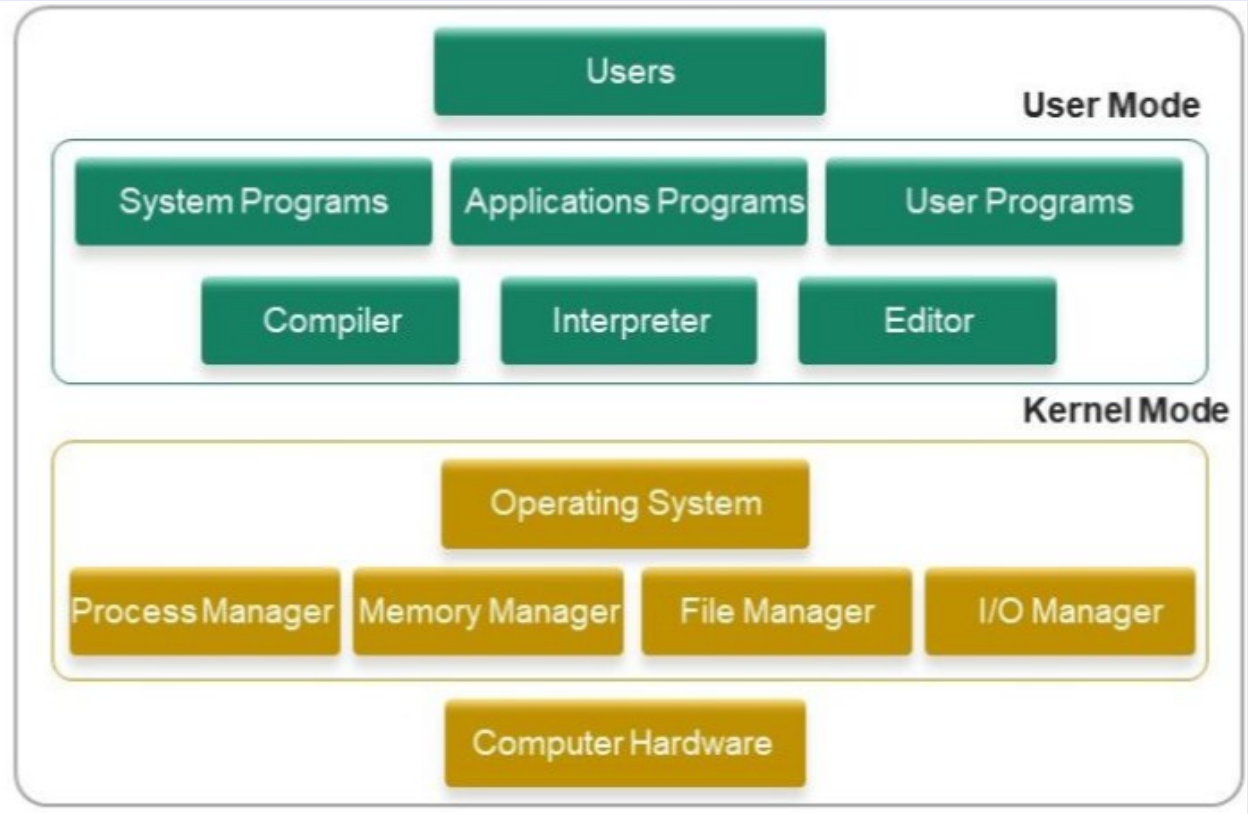
Interrupciones

- Segundo nivel de atención de Interrupciones
  - atender la Interrupción
- ¿Se puede interrumpir una interrupción?

Modos de la CPU (3)

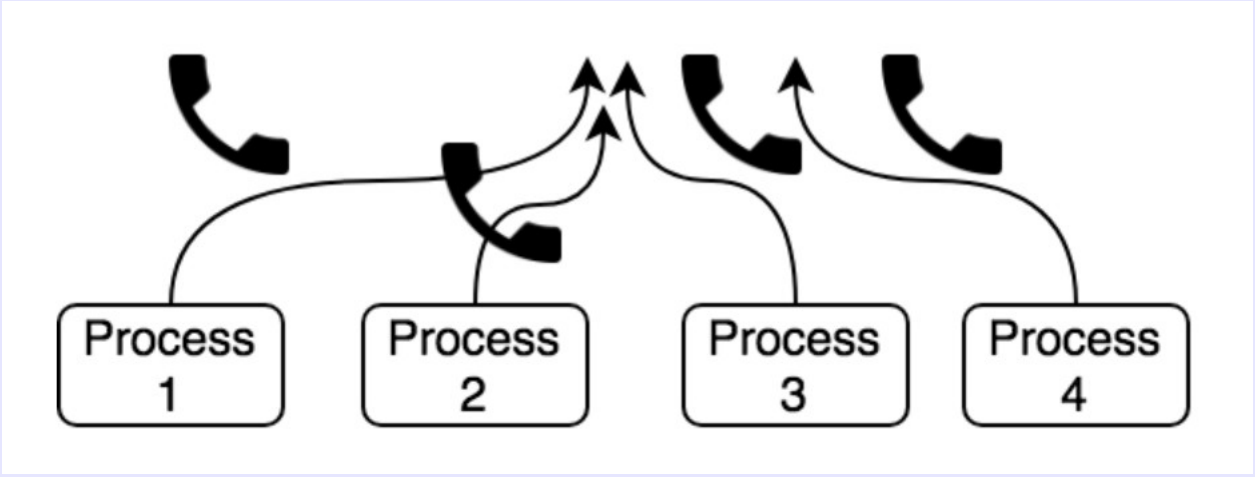
- Algunas arquitecturas incluyen más modos
  - X86 Modo real, protegido y virtual.
  - Modo hypervisor.
- Un sistema operativo puede tener partes corriendo en cada uno de los modos.
- Un programa de usuario solo corre en user mode.

Modos del Sistema Operativo



- Modo Usuario – userland.
- Modo Kernel – La CPU puede estar en algún modo user, kernel (supervisor), hypervisor, etc.

System Calls



Pedidos de servicio al sistema operativo.

- UNIX: sección 2 de *man*
- Rastreadas durante la ejecución con *strace (1)*
- Ponen al Sistema Operativo en modo Kernel

Pueden pasar a la CPU a modo Kernel (p. ej. *read()*)
Pueden no precisar hacerlo (p. ej. *getpid()*)
Depende del Sistema Operativo (y del autor del libro)

Library Calls

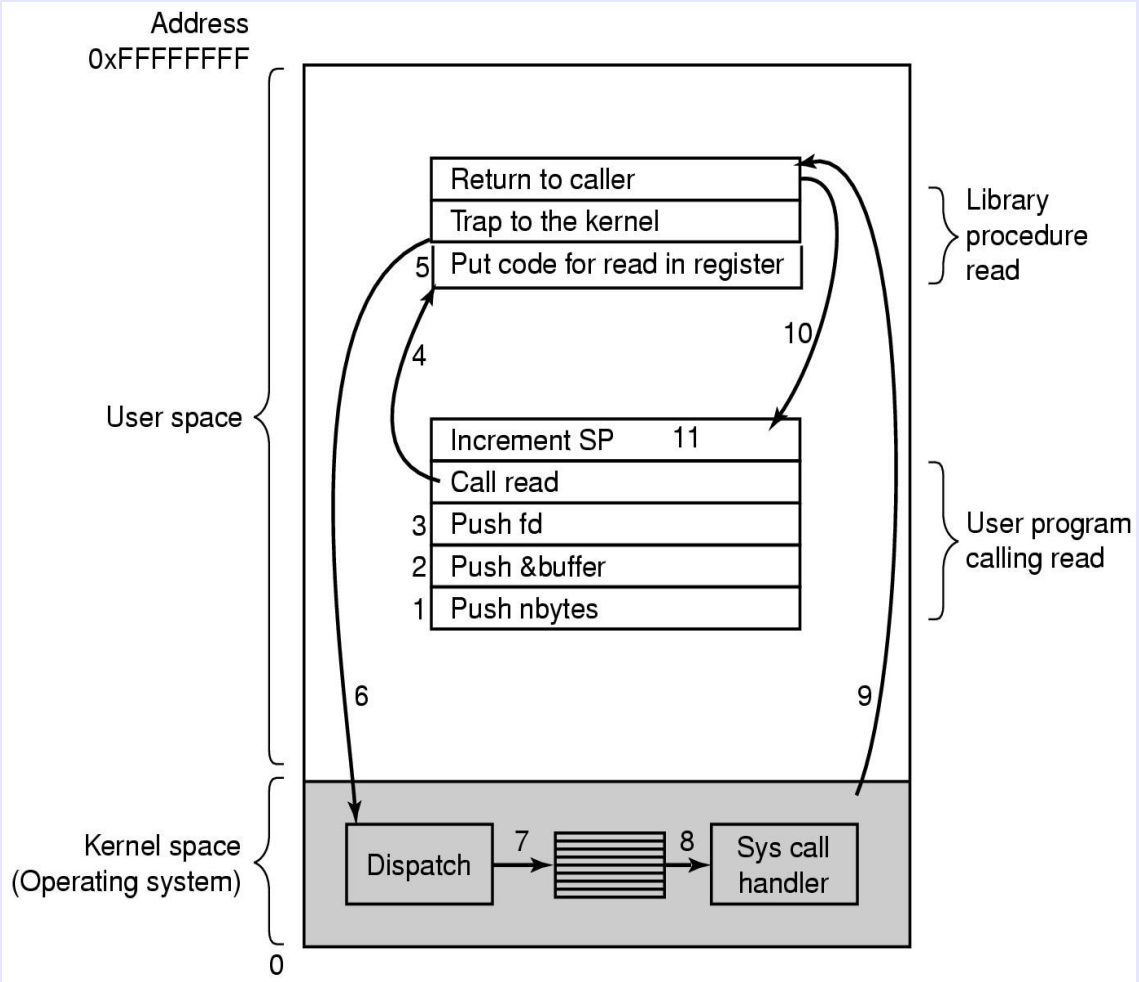
- Son llamados a procedimientos de biblioteca.
  - A veces terminan en System Calls (*printf( )*).
  - A veces no (*strcmp()*).

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

Library = Biblioteca (se consultan libros)
Bookstore = Librería (se venden libros)

Pasos para atender un Library Call con System Call y con trap

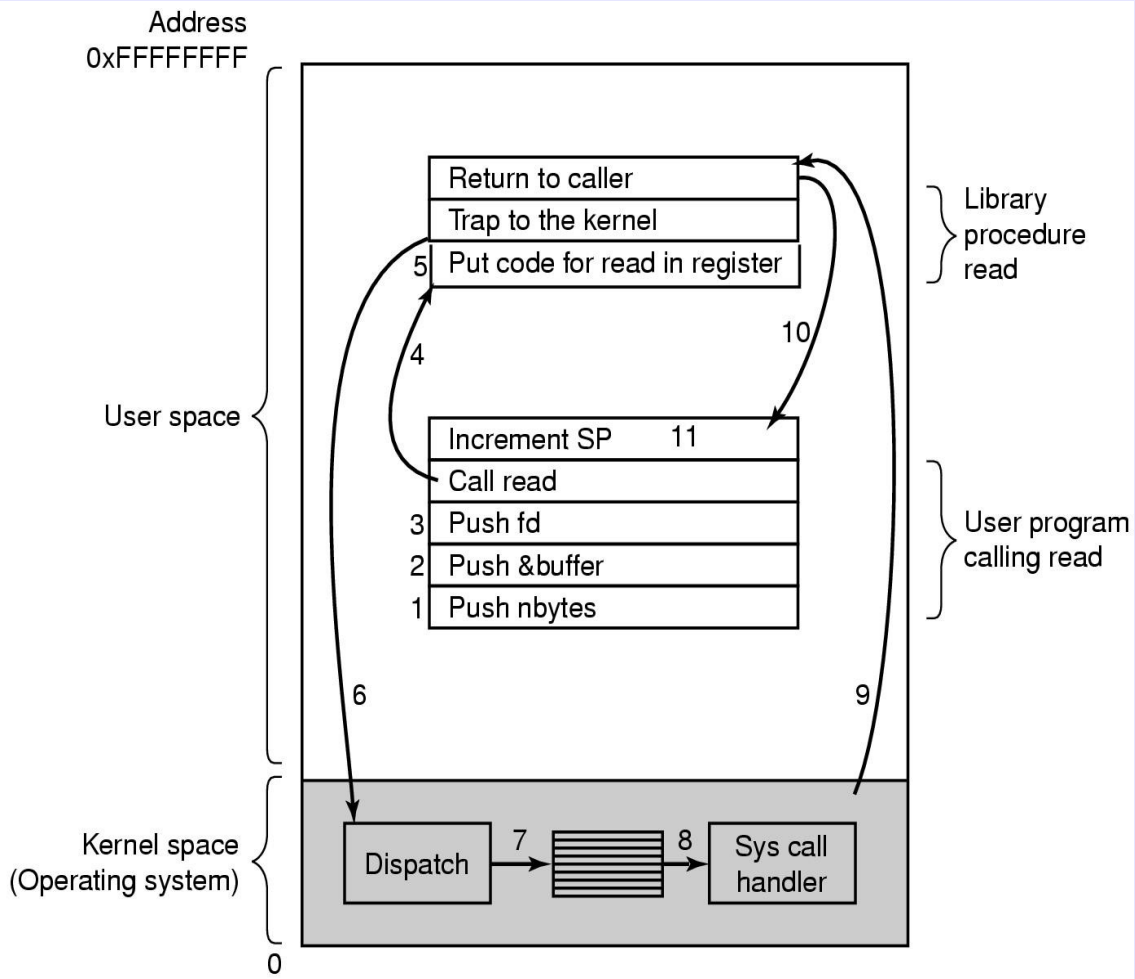
```
read (fd,&buffer,nbytes)
```



- 1, 2, 3: parámetros al stack.
- 4:invocación a función de biblioteca
- 5:escribe el valor del system call en el registro, que es donde el Sistema espera encontrarlo
- 6:Interrupción por Software y ejecución de primer nivel de atención. El código del "despachador" de interrupciones se encuentra en una dirección fija de memoria
- 7:Dependiendo del valor del system call se lanza el código de atención específico
- 8:Ejecución del system call
- 9:Devolución a próxima instrucción
- 10: Limpia Stack

Pasos para atender un Library Call con System Call y con trap

```
read (fd,&buffer,nbytes)
```



- 1, 2, 3 y 4 corresponden a un llamado convencional a un procedimiento.
- 5: el Sistema Operativo habilita el acceso a la memoria del Sistema Operativo (pasa a modo Kernel).
- 6 el procesador pasa a modo protegido.
- 7 y 8 se ejecutan en modo protegido del procesador.
- En 9 se vuelve a modo usuario del procesador, pero con el SO en modo Kernel.
- El paso 10 restablece la protección de memoria del SO (sale de modo Kernel).

Algunos System Calls (Unix)

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

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

Algunos System Calls (Win 32)

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

System Calls Win10