Porque es necesario un Sistemas Operativos

- **Como una máquina extendida**
  - La función del sistema operativo es presentar al usuario el equivalente de una **máquina extendida** o **máquina virtual** que es más fácil de programar que el hardware subyacente

- **Como gestor de recursos**
  - Asegurar un reparto ordenado y controlado de los procesadores, memorias y dispositivos de E/S, entre los diversos programas que compiten por obtenerlos

Conceptos de Sistemas Operativos

- **Proceso**: Programa en ejecución. Posee un espacio de direcciones propias del proceso. El espacio de direcciones contiene el programa ejecutable, sus datos y su(s) stack(s). Cada proceso posee además un conjunto de registros.

- **Thread**: Hilo de ejecución dentro de un proceso.

- **Scheduler**: Componente interno del S.O. encargado de la distribución de tiempo del procesador (o procesadores) entre los diferentes procesos/threads que compiten por ese recurso. En su operación, el scheduler puede quitarle la CPU a uno, para otorgárselo a otro.

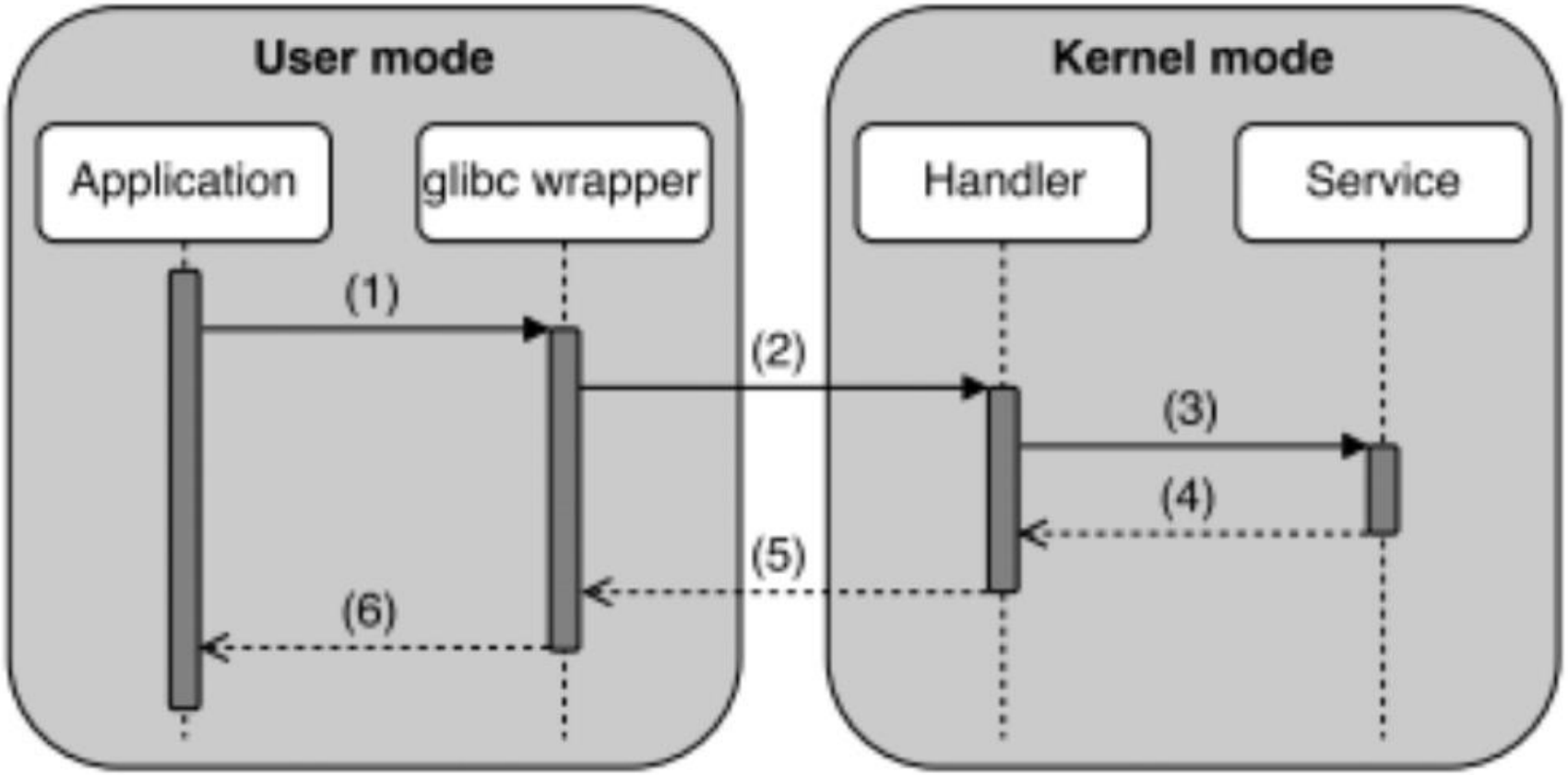
Modos de operación de una CPU

- En general, las CPUs tienen dos modos principales de ejecución: Kernel y Usuario. Está definido por un conjunto de bits en el PSW.
  - **Kernel Mode**: Pueden ejecutar cualquier instrucción, incluyendo el acceso al hardware. Un crash en kernel mode puede ser catastrófico
  - **User Mode**: Pueden ejecutar solo un subconjunto del set de instrucciones. No pueden cambiar los bits de modo del PSW. Debe delegar en funciones de sistema para el acceso al hardware.

System Calls

- Para obtener un servicio del sistema operativo, el programa de usuario debe hacer una llamada al sistema, la cual realiza un trap dentro del kernel e invoca al sistema operativo.
- La instrucción TRAP cambia de modo usuario a kernel y cede el control al sistema operativo.
- Una vez completado el trabajo solicitado, se devuelve el control al programa de usuario justo en la instrucción siguiente al llamado al sistema.
- Existen otros TRAPs: interrupciones, excepciones, reset, etc.

System Calls



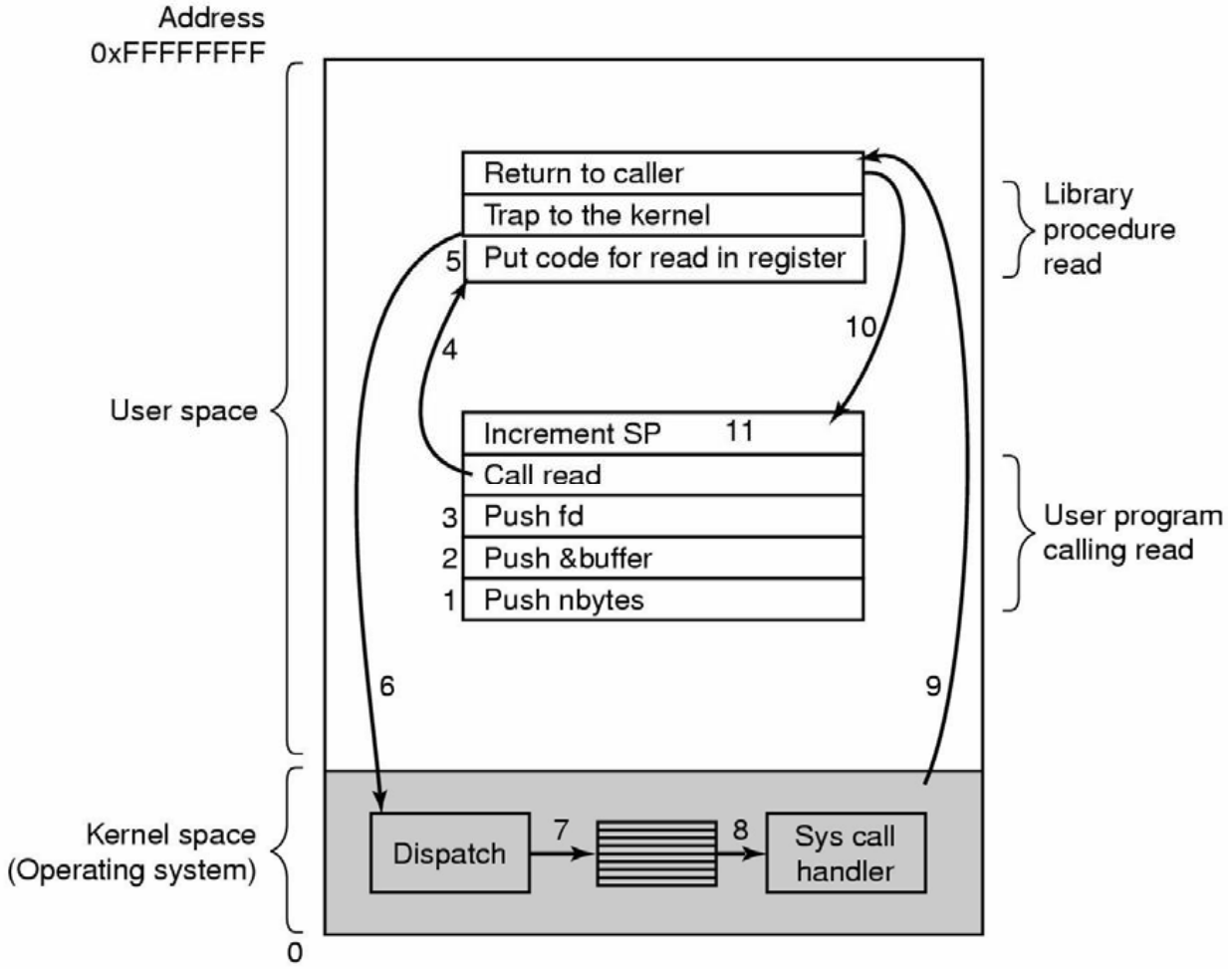
System Calls

- La interfaz entre el sistema operativo y los programas de usuario está definida por el conjunto de llamadas al sistema ofrecidas por el S.O. Varían de un S.O. a otro, aunque los conceptos subyacentes son similares.

- Ejemplo de un system call:

```
bytesRead = read(fd, &buffer, nBytes);
```

System Calls




```
(0) bytesRead = read(fd, &buffer, nbytes);
```

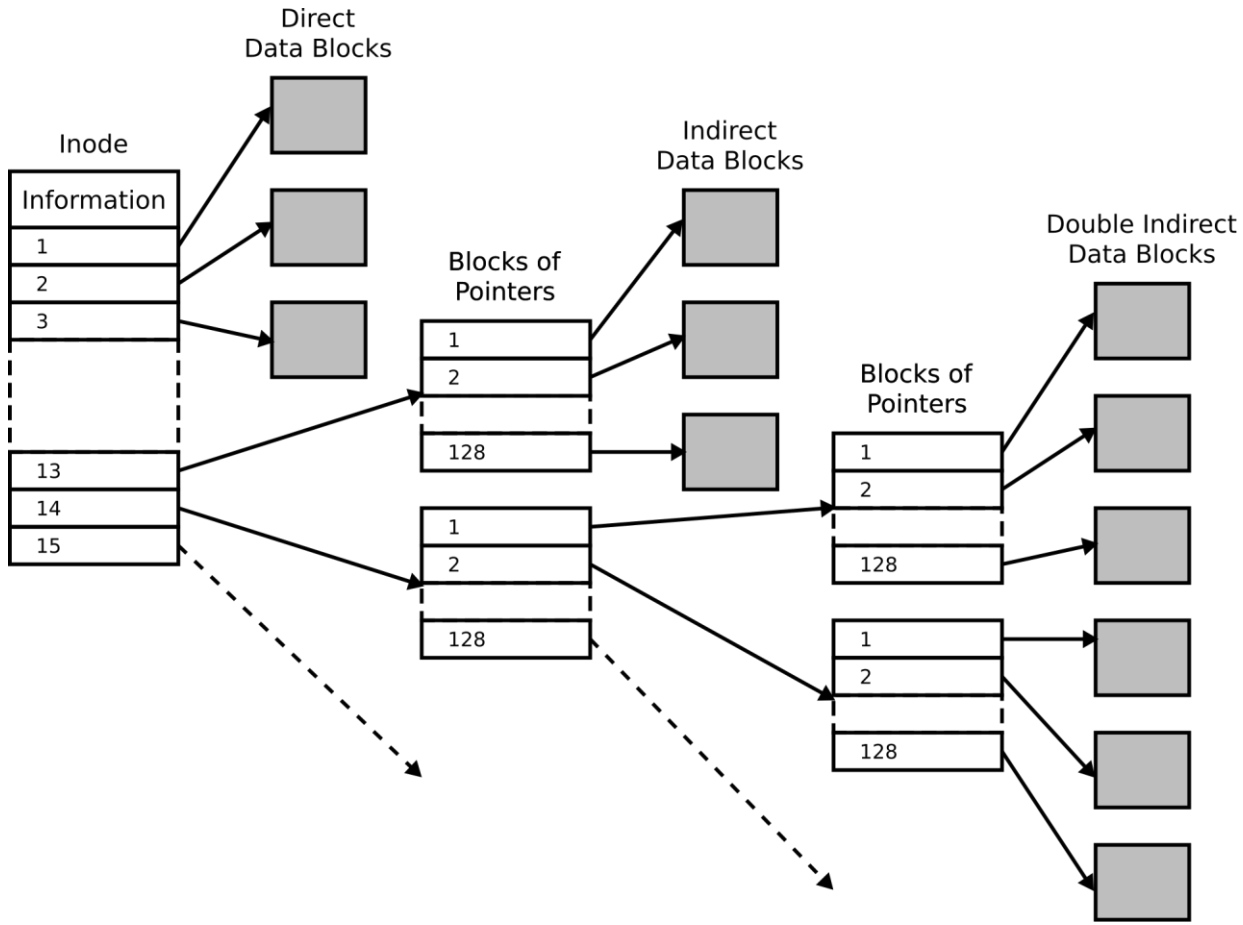
File System

- Estructura para guardar y recuperar información, en dispositivos de almacenamiento.
- Concepto de archivos y directorios.
- Estructura de árbol.
- Muchos tipos de filesystems:
  - ext2/3/4
  - NTFS
  - FAT12/16/32/ExFAT
  - ISO9660/RR extensions/Joliet
  - nfs / CIFS
- Comparación de filesystems

File System - Unix

- Ínodo y bloques de datos. Árbol único.
- El ínodo tiene toda la metadata del archivo.
- El nombre de un archivo no está en el ínodo. Está en los directorios, que no son más que archivos que tienen como dato un mapa de nombre a ínodo.
- Los archivos son “bag-o-bytes”. No tienen estructura.
- Existen archivos especiales:
  - Named fifos, sockets
  - Archivos de dispositivos. Bloques y caracteres.
  - Links simbólicos/hard links

Ejemplo de ínodo



man pages

```
man man
man [section] page
```

Section	Description
1	General commands
2	System calls
3	Library functions, covering in particular the C standard library
4	Special files (usually devices, those found in /dev) and drivers
5	File formats and conventions
6	Games and screensavers
7	Miscellanea
8	System administration commands and daemons