Tema 4:
Introducción a los Sistemas Operativos

Fundamentos de Computadores

Curso 2018/19

Índice (1/2)

Índice (2/2)

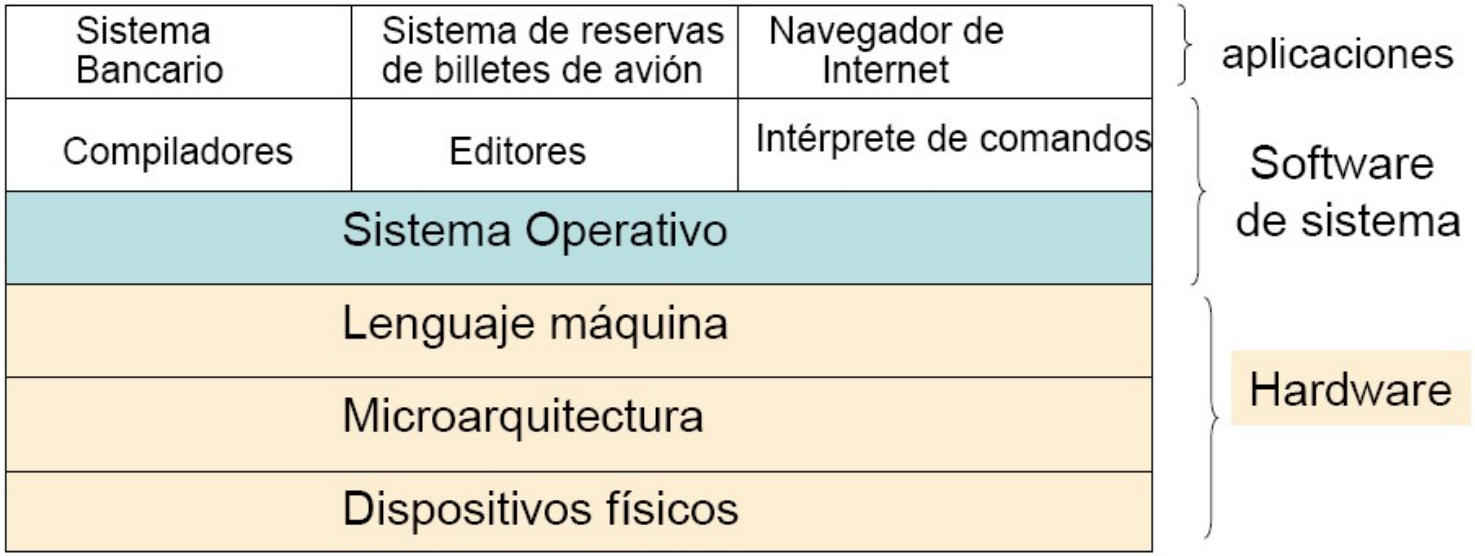
Índice

Concepto de SO

- Sistema Operativo:
  - Capa *sw* inmediatamente por encima del *hw* que...
    - Administra CPU(s):
      - Comparte CPU entre distintos procesos y núcleo del SO.
    - Administra memoria:
      - Comparte ésta entre procesos y núcleo.
      - Protege accesos cruzados indebidos.
    - Administra dispositivos:
      - Comparte éstos entre procesos y núcleo.
      - Protege accesos cruzados indebidos.
      - Ofrece interfaz uniforme y sencillo a muy distintos tipos de dispositivos (discos, teclado, pantalla, red, etc.).
  - Ejemplos:
    - Windows (2000, XP, Vista, ...), Linux (Ubuntu, Fedora, ...), Unix, FreeBSD, MacOS, ...
    - Diferencia núcleo (SO propiamente dicho) / distribución (incluye sw de sistema adicional).

Ubicación del SO en los sistemas de cómputo

- El SO se sitúa entre el hw "desnudo" y las aplicaciones:
  - Proporciona el interfaz adecuado para evitar la dificultad de tener que acceder directamente al hw.



Visiones de un SO

- Dos visiones complementarias de un SO:
  - Como máquina extendida, o virtual:
    - "Enmascara" la complejidad del *hw.*
    - Ofrece un uso más sencillo del computador:
      - Para usuarios: Programas de sistema (interfaces gráficos, intérpretes de comandos, editores, navegadores, sistemas de ficheros).
      - Para programadores: Llamadas al sistema (subrutinas de fácil uso para manejar memoria, procesos, ficheros, etc.).
  - Como administrador de recursos:
    - Controla y coordina que todos los recursos de la CPU se manejen de forma eficiente:
      - CPU.
      - Memoria.
      - Dispositivos.

Tipos de SSOO

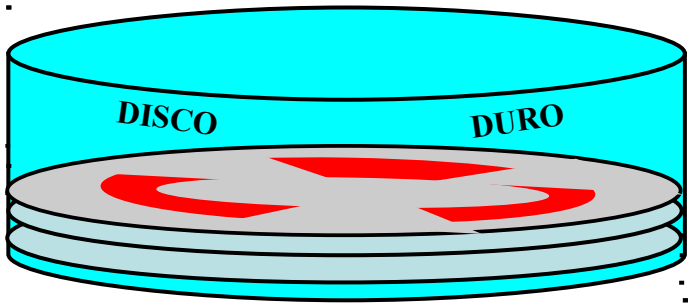
- Diversos tipos de SO:
  - Dependiendo de:
    - Potencia/características de la máquina.
    - Propósito principal del sistema de cómputo.
  - SO de **propósito general** para computador personal (ofimática, uso internet, juegos, software variado, etc.).
  - SO de **servidores** (servicios web, bases de datos, etc.).
  - SO de **tiempo real** (control industrial, robots, etc.).
  - SO **integrados** (dispositivos móviles, sistemas embebidos, etc.).
  - SO de **tarjeta inteligente** (funcionalidades mínimas, almacenamiento seguro de datos, etc.).
  - SO de **supercomputadores** (clusters, grandes máquinas, etc.).
  - Etc...

Conceptos clave en un SO

- Conceptos clave en un SO (1/3):
  - **Usuario**: persona que trabaja en el sistema; puede haber varios trabajando simultáneamente.
  - **Sesión**: Periodo de tiempo durante el que un usuario interactúa con el sistema.
  - **Programa**: Código ejecutable almacenado en disco, que eventualmente puede ser cargado en memoria para ser ejecutado.
    - Es un *concepto estático* (sin evolución ni estado).
  - **Proceso**: Programa en ejecución, que necesita recursos (CPU, memoria, ficheros, dispositivos, etc).
    - Es un *concepto dinámico*, con estado cambiante: contador de programa con siguiente instrucción a ejecutar, estado actual de los datos en memoria o disco, etc.
    - Es la unidad de trabajo del SO, que debe crear / eliminar / suspender / reanudar / comunicar / sincronizar los distintos procesos en ejecución.

Conceptos clave en un SO

- Conceptos clave en un SO (2/3):
  - **Fichero**: Unidad lógica de almacenamiento para datos que necesitan persistencia (discos, memorias USB, etc.).
    - Es una secuencia de octetos con un formato determinado, válida para cualquier clase de datos (texto, enteros, reales, imágenes, sonidos, vídeos, programas, librerías, formatos específicos de distintos tipos de *sw*...).



Sectores por los que se distribuye cierto fichero

El SO utiliza tablas de acceso para poder localizar los sectores donde se ubican los datos de los correspondientes ficheros, así como listas (recursivas) de nombres de archivos/subdirectorios para organizar jerárquicamente el sistema de archivos (estructura de directorios).

Conceptos clave en un SO

- Conceptos clave en un SO (2/3):
  - **Programas del sistema**: Programas para operaciones básicas que suelen acompañar al SO:
    - Manipulación de ficheros (crear, eliminar, ver, imprimir, etc.)
    - Intérpretes de comandos.
    - Proporcionar información de estado (procesos, memoria, disco).
    - Aplicaciones básicas (edición texto, gráficos, cálculo, etc.)
    - Utilidades de programación (compiladores, manejo librerías, depuradores, etc.).
    - En realidad, como procesos se les trata exactamente igual que a otras aplicaciones.

Conceptos clave en un SO

- Conceptos clave en un SO (3/3):
  - **Interfaz de usuario**: Permite al usuario dar órdenes al sistema. Hay de dos tipos:
    - Gráfico (GUI, Graphical User Interface): usa ratón, ventanas, iconos, menús, atajos de teclado, etc.
      - Presentan a los usuarios una visión sencilla e intuitiva del sistema.
      - Basados en un gestor de ventanas que permite arrancar y terminar aplicaciones, trabajar simultáneamente con varias actividades, manipular ficheros y directorios, configurar el sistema, etc.
    - De línea de comandos: usa un terminal con órdenes tecleadas, con diversos parámetros y opciones
      - Los intérpretes de comandos pueden llamar bien a **órdenes internas** (reconocidas y ejecutadas por el propio intérprete) o bien a **programas externos** (almacenados en su propio fichero ejecutable, y que el programa intérprete simplemente busca en disco, carga en memoria para ejecutar cuando se invocan).

Introducción a Linux

- Características principales de Linux (1/2)
  - Clon de Unix, iniciado por Linus Torvalds (1991).
  - *Open source*: Código fuente disponible, licencia GPL (puede usarse, modificarse y distribuirse libremente). Torvalds dirige la evolución de la rama principal.
  - Multiplataforma: corre en CPUs Intel, AMD, PowerPC, ARM, etc., de 32/64 bits. Muy portable por estar casi íntegramente escrito en C.
  - Multiusuario/multitarea (también aprovecha arquitecturas *SMP* y *multicore*).
  - Protección máxima entre procesos (memoria virtual, con espacios de direccionamiento independientes).
  - Soporte para múltiples sistemas de ficheros: ext4 (nativo), FAT (12, 32, VFAT), NTFS, ISO9660 (CD), UDF (DVD), ...
  - Soporte para múltiples protocolos de red: TCP/IP (Internet), Samba, NetBEUI (Windows), IPX (Novell), ...
  - Soporte para infinidad de dispositivos (impresoras, discos, tarjetas de red, *webcams*, etc.).

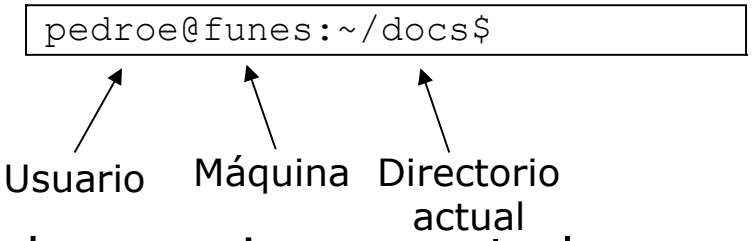
Introducción a Linux

- Características principales de Linux (2/2)
  - Múltiples distribuciones (Ubuntu, Fedora, Debian, SUSe, ...). Muchas con posibilidad de LiveCD.
  - Distribución = núcleo + software variado + sistema configuración (paquetes, admin., …).
  - Miles de aplicaciones disponibles (libres y comerciales):
    - Ofimática: LibreOffice (suite tipo Office) , lectores de PDFs, ...
    - Internet/comunicaciones: Chrome, Firefox, Konqueror, Thunderbird, Skype, ...
    - Gráficos: GIMP, Inkscape, ...
    - Multimedia: Mplayer, Amarok, VLC, Brasero, K3B, ...
    - Desarrollo: QtCreator, Eclipse, GDB...
    - Computación Científica: Matlab, Mathematica, Octave, ...
    - Otros: Emuladores (wine, vmware, virtualbox, ...), utilidades varias, juegos, ...

Índice

Acceso al sistema. Usuarios.

- Primeros pasos con Linux:
  - Acceso: *login* + *password*.
    - Posibilidad de múltiples usuarios y grupos.
  - Tareas administración: usuario *root*.
    - Único con acceso total al sistema.
  - Prompt del sistema:
    - Configurable, con información útil:



  - Listado de usuarios conectados:
    - Comandos `w`, `who`, `whoami`.
  - Salir de una sesión:
    - Comando `exit`.

Interfaces gráficas

- Interfaces gráficas en Linux (1/2):
  - X-Window: GUI gráfico que permite lanzar y manipular procesos mediante la metáfora del sistema de ventanas, manipulables con ratón y teclado.
  - X-Window es en realidad un proceso "servidor", al que se conectan las aplicaciones gráficas "cliente".
  - Aunque normalmente cliente y servidor se ejecutan localmente, el diseño orientado a la red de X-Window permite ejecutar aplicaciones gráficas remotas (en otra máquina), siempre que en nuestra máquina local se esté ejecutando un *servidor X*.
    - Obsérvese el papel "invertido" de cliente y servidor respecto al tradicional enfoque en el que el servidor es remoto y el cliente local
  - El servidor X captura todos los eventos de teclado y ratón y los envía a la aplicación, al tiempo que se encarga de mostrar en pantalla la salida gráfica de la misma en la(s) ventana(s) correspondientes.
  - startx: comando que arranca un servidor gráfico
    - Aunque la mayoría de las distribuciones lo arrancan automáticamente al encender.

Interfaces gráficas

- Interfaces gráficas en Linux (2/2):
  - El principal cliente de todo servidor *X-Window* es el gestor de ventanas, que determina la apariencia del escritorio y las ventanas.
  - En Linux son muy populares sobre todos dos entornos de escritorio: KDE y GNOME.
    - Pero existen otros muchos (p.e. más ligeros, especializados para *netbooks*, etc.).
    - KDE y GNOME son verdaderos entornos integrados, con cientos de aplicaciones de todo tipo (al estilo de los Windows XP, Vista, etc.)
    - Dado que todos funcionan sobre las X, sus aplicaciones son perfectamente compatibles.
  - Ctrl-Alt-F1 a Ctrl-Alt-F6 cambia a cualquiera de los 6 terminales de texto que por defecto lanza Linux al arrancar.
  - Ctrl-Alt-F7 vuelve al entorno gráfico (se pueden abrir más servidores, accesibles con Ctrl-Alt-F8, ... , pero no es muy útil en principio).

Intérprete de comandos

- ¿Por qué la línea de comandos?
  - Muy flexible y potente:
    - Todo programa se puede llamar desde ella.
      - Se le puede pasar mucha información al programa llamado (opciones, parámetros, ...)
    - Se pueden combinar comandos para hacer cosas muy potentes:
      - P.e. lo siguiente encuentra todos los ficheros fuente en C que cuelgan (recursivamente, a cualquier profundidad) del directorio "progs", e informa del tamaño total sumado por los mismos:

        ```
        find progs -iname "*.c" -printf " + %s" | xargs expr 0
        ```

        (más adelante detallaremos todo lo que hace que este comando funcione: tuberías, opciones del find, xargs, expr, etc.)
    - El propio intérprete de comandos (*shell*) es en realidad un potente lenguaje de programación, con bucles, funciones, arrays, que permiten hacer programas muy complejos:
      - P.e. gran cantidad de los comandos del sistema en `/bin` o `/usr/bin` son *scripts* de `bash` (`bash` es el nombre del intérprete de comandos particular que usaremos en Linux).

Intérprete de comandos

- Estructura general de una orden:
  - Orden [opciones] [parámetros]
  - Ejemplo: `ls -l -i fich1.txt fich2.txt` = `ls -li fich1.txt fich2.txt`
- Movimiento por árbol de directorios:
  - Comando `cd [ruta]`.
  - Comando `pwd`: Devuelve directorio actual.
  - Utilidades: tabulador completa ruta, `cd -` vuelve a directorio anterior
- Atajos de teclado:
  - Tabulador no sólo completa ruta, también comandos.
  - Ratón selecciona texto a copiar, botón central pega.
  - Historial de órdenes:
    - Flechas cursor nos mueven por él.
    - Consulta con comando `history`.
    - `Ctrl-R` busca comando anterior.
    - `!orden` ejecuta última orden (incluyendo opciones y argumentos).

Documentación y ayuda

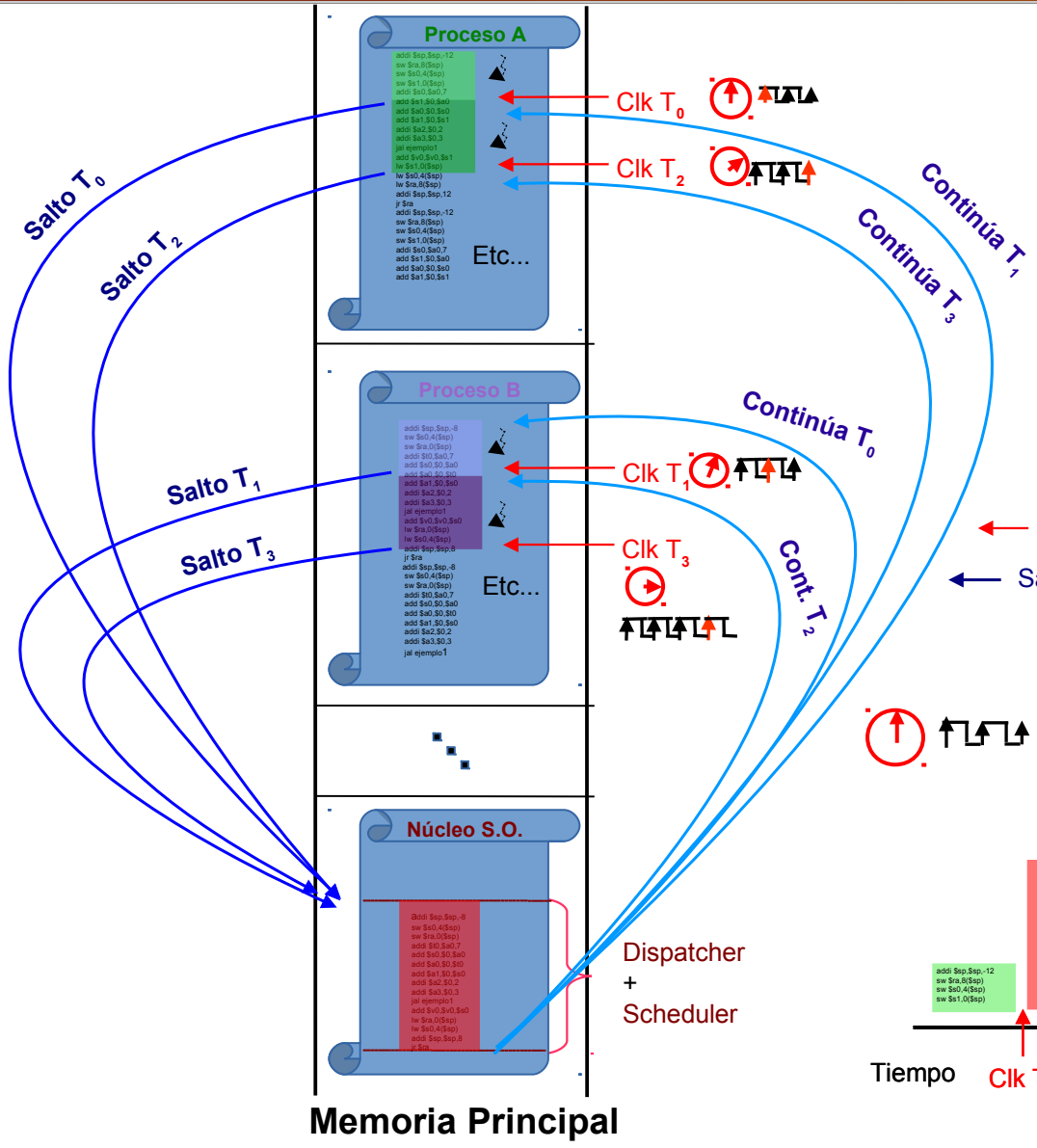
- Ayuda en Linux:
  - Orden `man comando` (comandos externos).
  - Orden `help comando` (comandos internos bash).
  - Otras posibilidades: comando `info`, flags `-h -help -?`, `/usr/share/doc/`, Google, ...
  - También muy útil y cómodo: en el `konqueror` (navegador de KDE), teclear `info:comando` o `man:comando` en la barra de navegación (p.e., `man:ls`).
- En general, todos los comandos que estudiaremos tienen multitud de opciones interesantes:
  - Aunque en clase estudiaremos sólo algunas de las más importantes ¡Es muy recomendable consultar el manual y experimentar con las demás!

Índice

Subsistema de gestión de procesos

- Los SO permiten la *multiprogramación*:
  - Uso compartido de la(s) CPU(s) entre distintos procesos
  - Naturalmente, es la vía para permitir múltiples usuarios.
  - Periódicamente (un elevado número de veces por segundo), un reloj interrumpe al proceso actual, y entra a ejecutarse código del kernel que *cambia el contexto* para ejecutar otro proceso.
  - La parte del núcleo encargada (planificador, o *scheduler*) es la responsable de optimizar el uso de la(s) CPU(s).
  - Los procesos bloqueados en una E/S (p.e., lectura de disco) ceden automáticamente la CPU a otro proceso (el planificador siempre intenta optimizar el uso de este recurso).
  - Todos los procesos van avanzando con sensación de simultaneidad, a pesar de que puede no haber paralelismo *real* (p.e., si hay sólo una CPU).

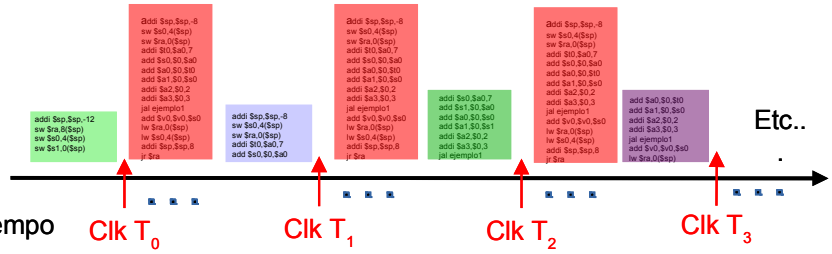
Subsistema de gestión de procesos



Clk T_i = Interrupción del Reloj

Salto T_i = Salto a la S.A.I. que provoca la Clk T_i

A la C.P.U le llega un pulso de reloj cada x milisegundos cediéndose el control de ejecución al S.O. que guarda datos del proceso actual para poder devolvérselos posteriormente.



Subsistema de gestión de procesos

- Mecanismos del SO para la gestión de procesos:
  - Creación (jerárquica) de procesos: todo proceso es creado por un proceso padre. La raiz de la jerarquía es el proceso inicial del sistema (*init*, en Linux).
  - Terminación de procesos: voluntaria (*exit*) o externa (p.e., *kill*, en Linux).
  - Monitorización de procesos:
    - Listado de procesos, dueño de cada proceso, porcentaje de uso de CPU, memoria, etc.
  - Comunicación:
    - Se autorizan intentos explícitos de comunicación entre procesos. P.e., tuberías:

```
$ ls -l | grep ".c" | wc -l
```

Subsistema de gestión de la memoria

- La RAM (memoria principal) es compartida por múltiples procesos simultáneamente (y por el propio *kernel*).
- El núcleo protege y aísla a unos procesos de otros:
  - Los ubica en compartimentos "estancos" de memoria (*espacios independientes de direccionamiento*)
  - Cada proceso tiene un "espacio de direcciones virtual" (p.e., siempre partiendo de 0x00000000, y dividido en partes de código, de datos y de pila).
  - El espacio de direcciones virtuales de cada proceso es mapeado por el sistema a distintas direcciones físicas.
  - Incluso, si es necesario, parte del espacio virtual de un proceso es mantenido en disco si no cabe en RAM (*memoria virtual*).
- El SO se encarga en todo momento de:
  - Controlar las zonas de memoria libres y ocupadas.
  - Asignar y recuperar espacio.
  - Mover datos y/o código a/desde RAM desde/a disco, según encuentre necesario.

Subsistema de gestión de la entrada/salida

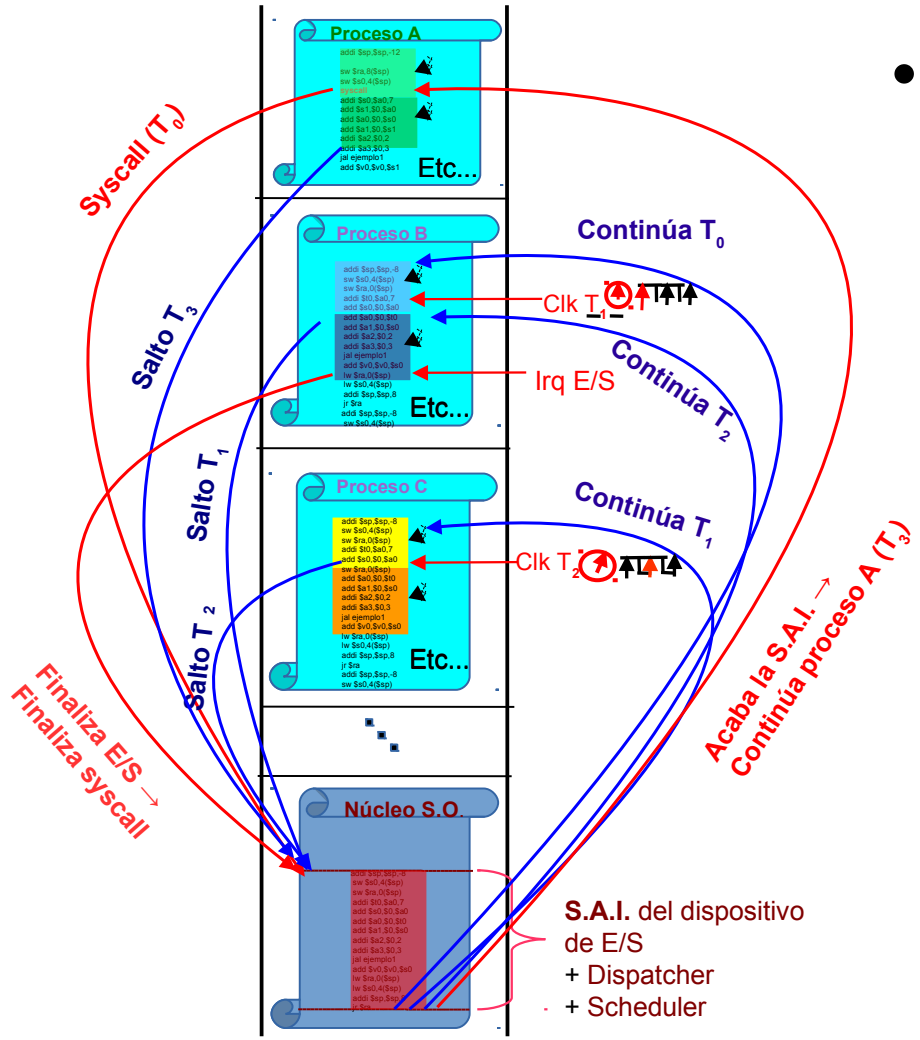
- Interfaz homogénea y sencilla a la E/S:
  - El SO oculta las particularidades de los distintos tipos de dispositivos.
  - P.e. el usuario ve un *sistema de ficheros* organizado jerárquicamente con carpetas y ficheros, y un determinado espacio libre, pero:
    - En realidad un disco es sólo un inmenso array lineal de sectores de (p.e.) 512 bytes cada uno.
    - El SO maneja todos los aspectos físicos de bajo nivel (espacio libre, asignación física de sectores a ficheros, organización de directorios, etc.), y...
    - ...ofrece una visión de alto nivel al usuario, con conceptos como dueños, permisos, tamaños de ficheros, fechas de creación/modificación/uso...
    - ...y utilidades para crear, eliminar y manipular ficheros y directorios (ya a nivel lógico).

Subsistema de gestión de la entrada/salida

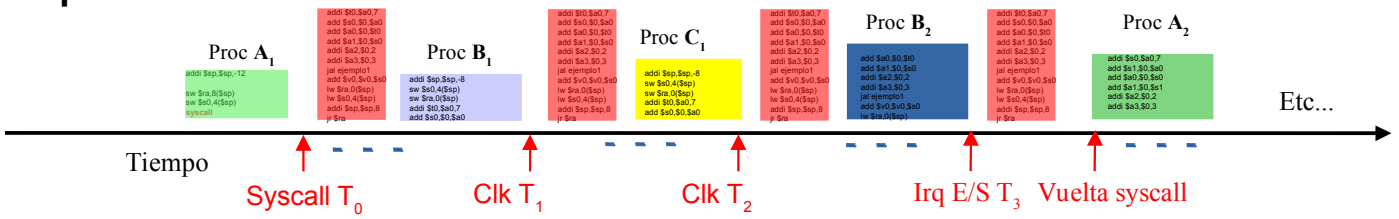
- Los dispositivos de E/S se manejan mediante:
  - **Controladora de dispositivo**:
    - Chip que controla físicamente el dispositivo.
    - Acepta comandos elementales y los ejecuta (p.e., mueve la cabeza del disco a tal posición, lee un sector en tal buffer, etc.).
    - La comunicación se realiza a través de determinados registros propios de la controladora.
  - **Manejador de dispositivo (driver)**:
    - Software que se comunica con la controladora.
    - Lo ejecuta el Kernel del SO en *modo privilegiado* (Uso no restringido del procesador).

Subsistema de gestión de la entrada/salida

- E/S asíncrona mediante interrupciones:
  - Cuando el manejador le pide a la controladora realizar una operación de E/S (p.e., leer un sector de disco), sabe que éste va a tardar un cierto tiempo.
  - Así que le pide también que genere una interrupción al terminar (a modo de aviso).
  - Entonces el núcleo suspende el proceso que espera la transacción (le quita la CPU y se la da a otro proceso).
  - Al finalizar la transferencia, la interrupción hw generada hace que el SO vuelva a tomar el control, y se despierta al proceso bloqueado.



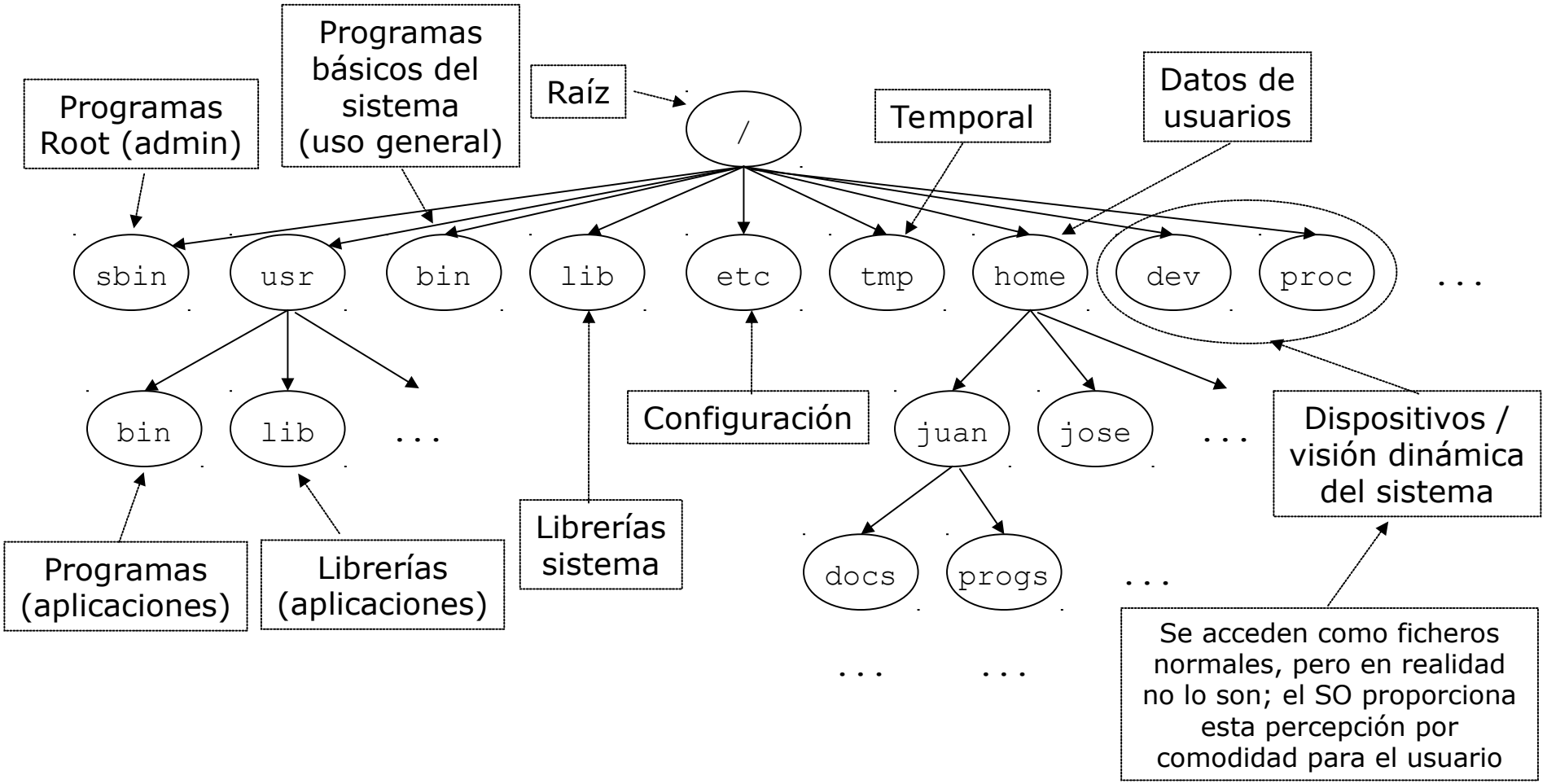
- Esquema de tratamiento de una llamada al sistema:
  1. Proceso A realiza llamada al S.O. (syscall).
  2. Núcleo del S.O. lanza orden de E/S al dispositivo, y como la respuesta va a llevar un tiempo, pasa el control a otros procesos (B, C) mientras el dispositivo acaba su tarea.
  3. Dichos procesos (B, C) pueden ir siendo a su vez interrumpidos por el planificador...
  4. … hasta que el dispositivo de E/S termina su tarea y lanza interrupción (irq).
  5. Núcleo entonces vuelve a tomar el control, y despierta al proceso bloqueado (A).



Índice

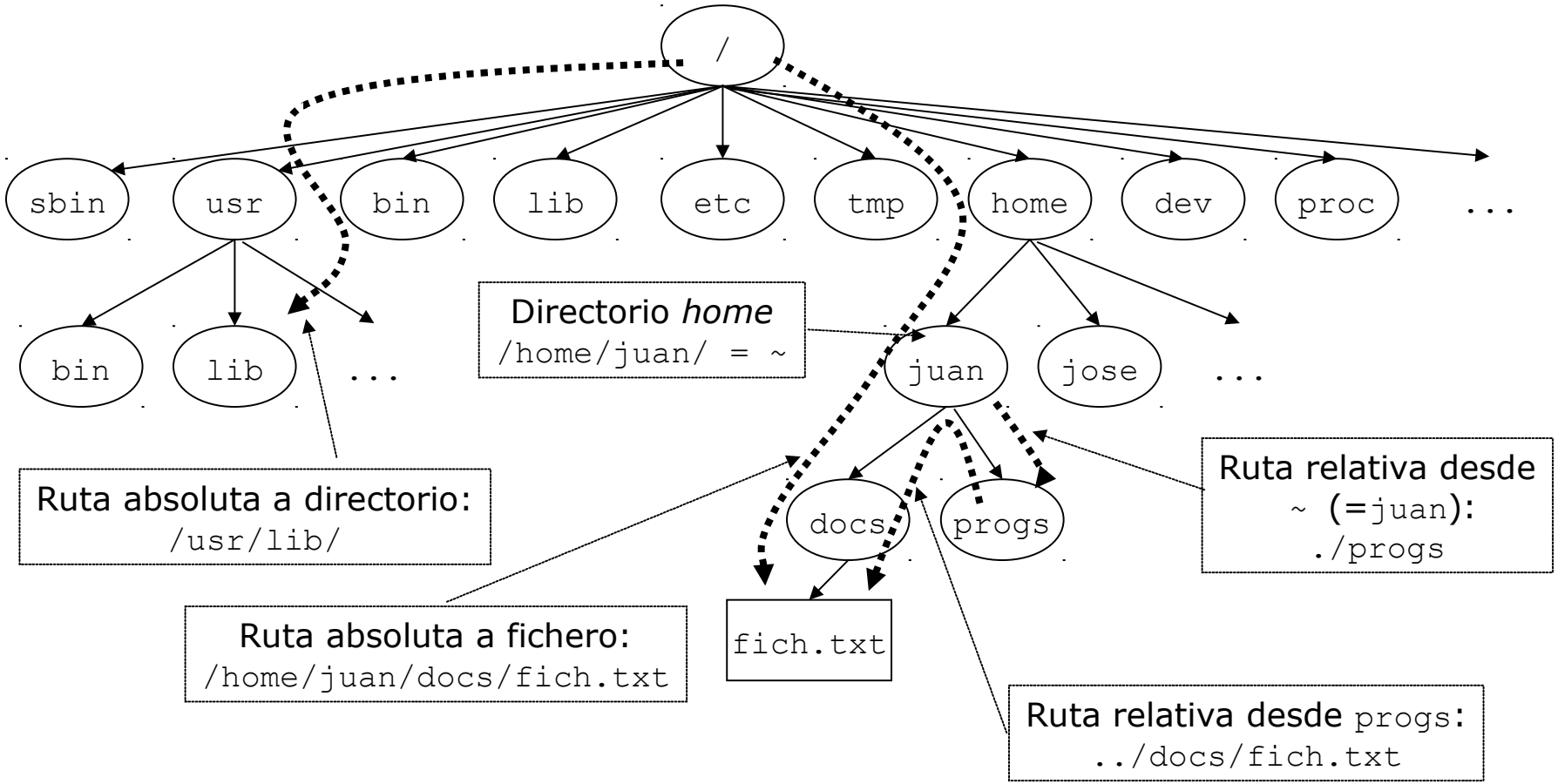
Jerarquía de directorios

- Jerarquía de directorios (1/2):
  - Cada directorio es una "carpeta" que puede contener ficheros u otros subdirectorios:



Jerarquía de directorios

- Jerarquía de directorios (2/2):
  - Rutas absolutas/relativas
    - Directorio actual: . Directorio padre: . . Directorio *home*: ~

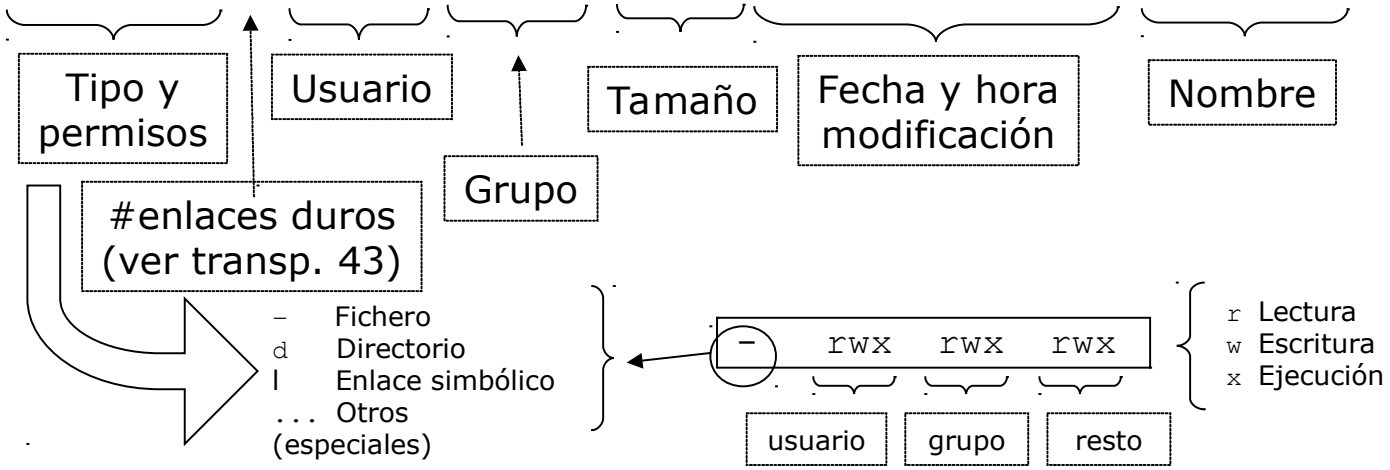


Consultas al sistema de ficheros

- Listado de ficheros:
  - Orden `ls [-R -d -a ...]`:

```
pedroe@funes:~/repositorios/fc-grad/trunk/docs/tema3$ ls -l

total 372
drwxr-xr-x 3 pedroe pedroe   4096 2009-03-31 14:12 fig
-rw-r--r-- 1 pedroe pedroe    246 2009-03-31 14:08 Makefile
drwxr-xr-x 3 pedroe pedroe   4096 2009-03-31 14:15 progs
-rw-r--r-- 1 pedroe pedroe 267776 2009-03-31 14:08 tema4.ppt
-rw-r--r-- 1 pedroe pedroe  90001 2009-03-31 14:08 tema4.tex
```



Consultas al sistema de ficheros

- Comodines:
  - Sirven para especificar varios ficheros a la vez. Pueden usarse en cualquier lugar de la línea de comandos. P.e., `ls -l *.c *.h`. Es el propio intérprete de comandos (`bash`) el que se encarga de realizar las sustituciones pertinentes:
    - `*` : Significa 0 o más caracteres cualquiera.
    - `?` : Un carácter cualquiera.
    - `[a-z]` : Cualquier carácter del rango indicado.
    - `[!b-d]` : Cualquier carácter que no esté en el rango indicado.
    - `{nom1,nom2,...}` : Cualquier secuencia de caracteres de la lista.
  - Ejemplos:
    - `*.{c,h}` : `pepe.c, f1.c, hola.h, antonio.h ...`
    - `{hola,adios}.[a-c]` : `hola.a, adios.a, hola.b, adios.b, hola.c, adios.c`
    - `*.?` : Todos los ficheros con extensión de un carácter. P.e. `pepe.h, f1.a, hola.c, …`
    - `*.[!c]` : Todos los ficheros con extensión de 1 carácter que no sea `c`. P.e. `pepe.h, f1.a`, pero no `fichero` o `antonio.c ...`

Consultas al sistema de ficheros

- Ficheros y directorios ocultos:
  - Su nombre comienza con el carácter `.` (punto).
  - No se muestran en listados `ls` normales; hay que usar la opción `ls -a` para verlos.
- Visionado de ficheros:
  - `cat fichero ...`: Muestra texto en terminal.
  - `less fichero ...`: Muestra texto en terminal, con posibilidad de *scroll* y búsqueda de palabras (con `/palabra`).
  - `hexdump [-C] archivo`: Vuelca el archivo en formato hexadecimal en el terminal.

Consultas al sistema de ficheros

- Búsqueda de ficheros:
  - `find directorio[s] expresiones`: Busca (recursivamente) en los directorios indicados entradas (fich. o dir.) que cumplan una serie de propiedades.
  - Ejemplos de opciones interesantes:
    - `find dir -type f -iname "*.tgz"`: Busca dentro de `dir` ficheros con extensión `.tgz` . (o `.TGZ`, o `.Tgz`, etc.).
    - `find dir -type d -name "dir_*"`: Busca en `dir` subdirectorios cuyo nombre empieze exactamente con la cadena `"dir_"`.
    - `find dir -size +140k -o -mtime -100`: Busca en `dir` subdirectorios de más de 140 KB, o bien que hayan sido modificados hace menos de 100 días.
    - `find . ! -user pedro -printf "%s %u %p\n"`: Busca en el directorio actual entradas que no sean del usuario `pedro`, y para cada una de ellas imprime una línea con el tamaño, el usuario y el camino completo al fichero.

Consultas al sistema de ficheros

- La opción `-printf` del find es muy versátil:
  - Está basada en la función `printf` (*imprimir con formato*) del lenguaje C, para imprimir cadenas de texto mezclado con otra información:
    - Ejemplo de C:

      `printf("La variable i contiene el valor %d \n",i);`

      Si `i=10`, imprime la línea (el `\n` final es retorno de carro):

      `La variable i contiene el valor 10.`
  - En el caso de la opción del `find`, los distintos valores de la cadena de formato (`%carácter`) indican distintos tipos de información útil para el fichero/directorio encontrado:
    - Ejemplos: `%s` = tamaño del fichero, `%u` usuario del fichero, `%p` = camino completo al fichero, etc.
    - Consultar "`man find`" para obtener la lista completa de posibilidades (muy amplia).
  - Ejemplo adicional:
    - `find . -type f -printf "El fichero %p ocupa %s bytes\n"`: Busca ficheros que cuelguen del directorio actual, y para cada uno de ellos imprime una línea de la forma "`El fichero ejemplo.txt ocupa 1239 bytes`".

Consultas al sistema de ficheros

- Resumen orden `find`:
  - Busca entradas (archivos, directorios, etc.) en el directorio (o directorios) indicado y en todos sus subdirectorios de forma recursiva.
  - Formato: `find <directorio(s)> <opciones>`
  - Opciones más interesantes:
    - `-name <nombre>`: nombre del fichero a buscar (se pueden utilizar comodines, en cuyo caso, se debería encerrar el nombre entre comillas).
    - `-iname <nombre>`: similar al anterior pero sin distinguir mayúsculas y minúsculas.
    - `-user <usuario>`: propietario al que debe pertenecer el fichero.
    - `-group <grupo>`: grupo al que debe pertenecer el fichero
    - `-type c|b|d|l|f`: tipo del fichero (carácter, bloque, directorio, enlace simbólico o fichero regular).
    - `-mtime [+|-]n`: fichero modificado hace más de, menos de o exactamente n días.
    - `-atime [+|-]n`: fichero utilizado hace más de, menos de o exactamente n días.

Consultas al sistema de ficheros

- Opciones más interesantes (continuación):
  - `-size [+|-]n`: fichero con un tamaño de más, menos o exactamente n bloques de 512 bytes.
  - `!`: negación de un criterio.
  - `-o`: O lógico de criterios.
  - `-a`: Y lógico de criterios (esta es la operación lógica implícita cuando se indican varios criterios).
  - Ejemplo combinación lógica:

    ```
    find /usr -type f -size +1k -o ! -user miguel
    ```

    Busca, a partir del directorio /usr, aquellos ficheros regulares cuyo tamaño sea superior a 1 KB o no pertenezcan al usuario miguel.

Manipulación del sistema de ficheros

- Creación, copia, movimiento, borrado, compresión:
  - `touch fichero ...`: Crea un nuevo fichero vacío, o si el fichero ya existía, le pone fecha y hora de última modificación la actual.
  - `cp fichero[s] fichero|directorio`: Copia un fichero a otro (o varios ficheros a un directorio). También funciona para copiar subdirectorios (opción `-r`).
  - `mv`: Similar a cp, pero para mover (o renombrar) ficheros.
  - `rm [-i -r] fichero[s] | directorio[s]`: Borrar ficheros o directorios (con posible confirmación y posible recursividad en directorios).
  - `tar` : compresión/descompresión:
    - `tar czvf result.tgz directorio[s] fichero[s]`: Empaqueta y comprime los directorios (recursivamente) y ficheros indicados en un solo archivo `.tgz` . Se guardan rutas relativas y permisos.
    - `tar tzvf result.tgz`: Lista en el terminal los contenidos del fichero `.tgz` .
    - `tar xzvf result.tgz`: Descomprime el fichero `.tgz` , recuperando los ficheros, permisos y la estructura de directorios original.
- Creación/borrado de directorios:
  - Creación de directorios: `mkdir nombredeldirectorio`
  - Borrado de directorios (si están vacíos; si no, usar `rm` con opción `-r`): `rmdir nombredeldirectorio`

Manipulación del sistema de ficheros

- Permisos:
  - `chmod octal fichero[s] | directorio[s]`: Cambia permisos; `octal` es un número de tres cifras en octal, que codifica los 3 grupos de 3 permisos:
    - P.e. 754 = 111 101 100 = `rwx r-x r—`
  - `chown nuevo_prop fichero[s] | directorio[s]`: Cambia el propietario de un fichero.
  - `chgrp nuevo_grupo fichero[s] | directorio[s]`: Cambia el grupo de un fichero.
- Significado permisos en directorios:
  - `r`: Permiso para leer el directorio (p.e. con `ls`, `find`).
  - `w`: Permiso para escribir en el directorio (p.e. Para crear y borrar ficheros en él).
  - `x`: Permiso acceder al contenido del directorio (p.e. con `cd` o con `find`).

Manipulación del sistema de ficheros

- **Enlaces**:
  - Se localizan en el sistema de archivos como ficheros/directorios normales, pero en realidad son sólo referencias (enlaces) a otro fichero/directorio en otro lugar del sistema de archivos.
  - Se crean con el comando `ln` (de *link*). Ejemplo:

    ```
    ln /datos/fichero.tgz /home/usuario/enlace.tgz
    ```

    Crea una entrada llamada `enlace.tgz` en `/home/usuario`, que apunta al fichero `/datos/fichero.tgz` (y éste sólo ocupa un lugar en el disco, NO está duplicado).
- **Enlace *simbólico* vs. *físico***
  - El ejemplo anterior es un enlace *físico*: después de crear el enlace, puede borrarse cualquiera de las entradas, que la otra permanecerá intacta. Sólo al borrar la última se borrará realmente liberando espacio en disco.
  - Si el enlace se hubiera creado *simbólico* (opción `-s`):

    ```
    ln -s /datos/fichero.tgz /home/usuario/enlace.tgz
    ```

    entonces los datos reales se almacenarán en el origen (`/datos/fichero.tgz`), y si se borra éste antes que el enlace, éste (`/home/usuario/enlace.tgz`) quedará "colgando" (apuntando a nada).

Discos

- Discos:
  - Sobre los discos "crudos" (/dev/sda, /dev/sdb, /dev/sdc, ...) se crean los sistemas de ficheros con "estructura" (directorios, ficheros con nombre, fechas, etc.).
    - **Nota**: `sd` → discos duros, USB, etc; `a`, `b`, `c`, ... → distintas unidades; `1`, `2`, `3` → distintas particiones.
    - Ejemplos: /dev/sda2, /dev/sdb1, ...
  - `df` : Informa de espacio total, libre y ocupado en todos los sistemas de ficheros montados (discos duros, disquetes, discos USB, etc.)

```
$ df
S.ficheros Bloques de 1K   Usado     Dispon    Uso%  Montado en
/dev/sda1  171058748       80443244  81994628  50%   /
...
```

  - `du [-h -s ...] [directorio]`: Muestra lo que ocupan realmente en disco el directorio indicado y todo lo que cuelga de él.

Índice

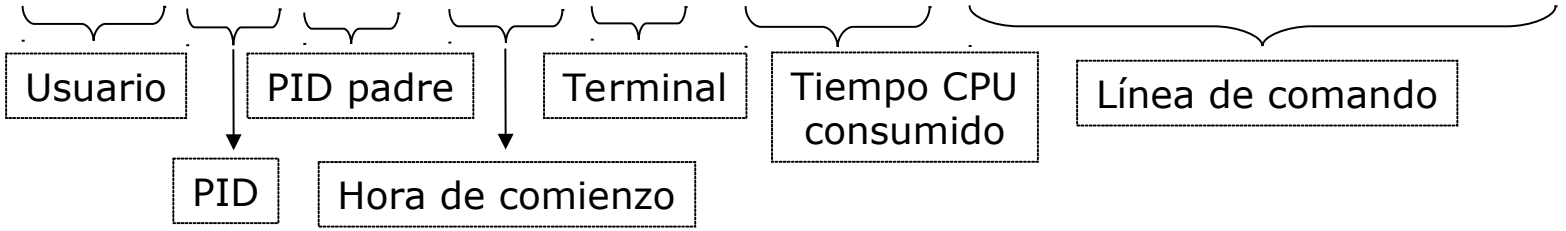
Lanzamiento y control de procesos

- Lanzamiento de procesos desde la línea de comandos:
  - `comando1 ; comando2 ; comando3 [intro]`: Lanza procesos en *primer plano* (hasta que uno no acaba, no se lanza el siguiente; y hasta que no acaba el último, no vuelve el *prompt* del `bash`).
  - `comando1 & comando2 & comando3 & [intro]`: Lanza procesos en *segundo plano* (cada uno no tiene que esperar a que termine el anterior, y el *prompt* vuelve al `bash` inmediatamente).
    - Ejemplo: `$ firefox &` : Lanza una ventana con el navegador, y pueden seguir tecleándose comandos.
    - El bash siempre imprime el PID (identificador de proceso) de cada proceso lanzado en segundo plano.
- Control de procesos desde la línea de comandos:
  - Con Ctrl-C se mata un proceso en primer plano.
  - Con Ctrl-Z se duerme un proceso en primer plano, que luego se puede despertar con `fg` (en primer plano) o `bg` (en segundo).
  - `Kill [-9] PID`: Lanza una señal para terminar al proceso identificado por PID, que lo mata (salvo si el proceso la "captura"). La opción –9 lo termina forzosamente (no es "capturable").

Monitorización de procesos

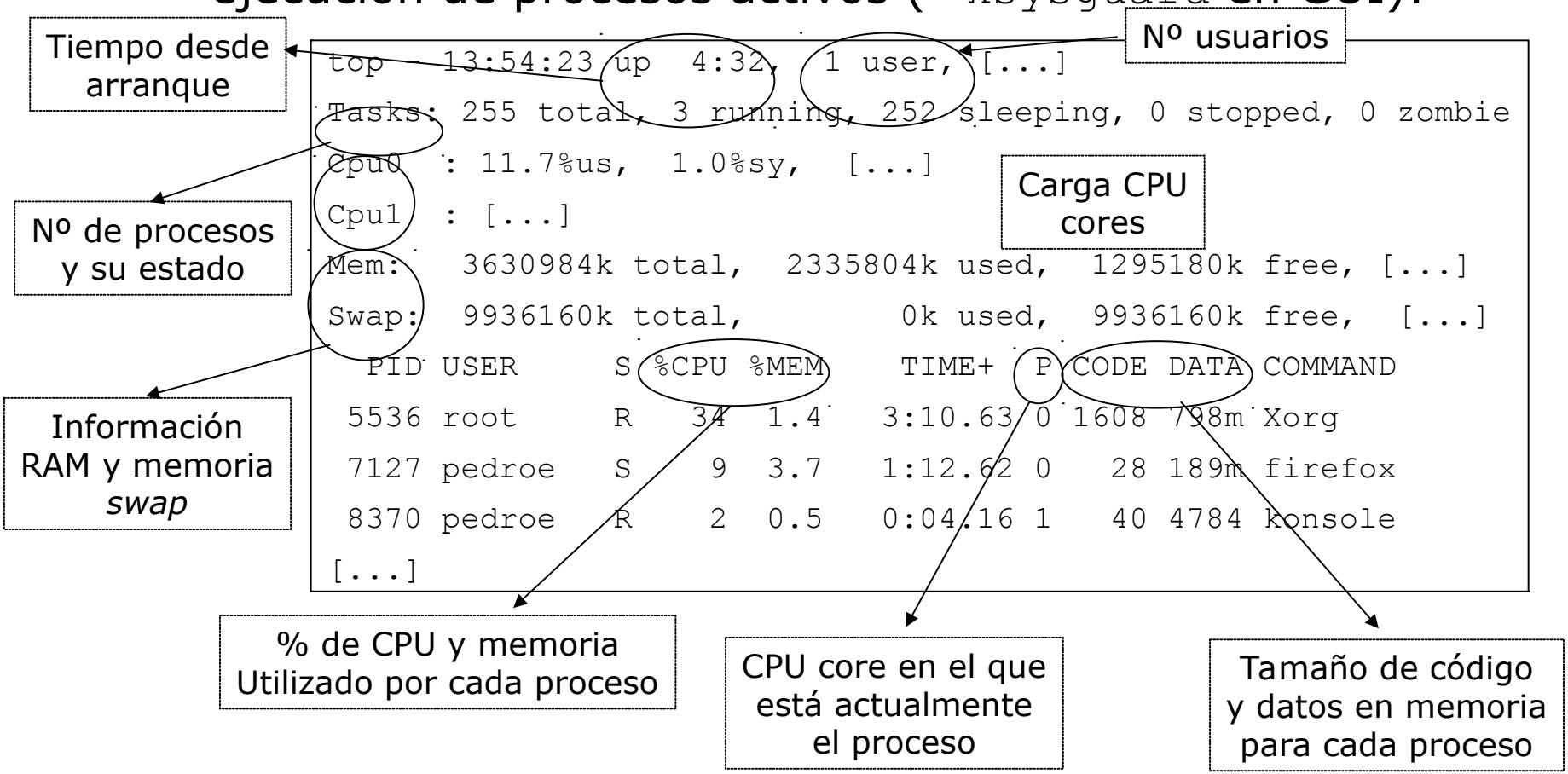
- Listado de procesos:
  - `ps [-Af]` : Muestra lista de procesos activos. Con las opciones indicadas, muestra todos los procesos del sistema, con distintas informaciones de interés:

```
$ ps -Af
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 09:21 ?        00:00:01 /sbin/init
[...]
root      5013     1  0 09:21 tty4     00:00:00 /sbin/getty 38400 tty4
[...]
pedroe    9420     1  0 12:56 ?        00:00:00 /usr/bin/mplayer video.avi
[...]
pedroe    9510  8874  0 13:38 pts/2    00:00:00 ps -Af
```



Monitorización de procesos

- Monitorización en tiempo real:
  - `top`: Muestra y actualiza constantemente información de ejecución de procesos activos (=`ksysguard` en GUI):



Monitorización de procesos

- Directorio virtual `/proc` :
  - Se ve como un subdirectorio del sistema de ficheros, pero es virtual (no está en disco, y se actualiza constantemente):
  - Se consulta como el sistema de ficheros normal (`cat` sobre ficheros, `cd` a directorios, etc.), pero en realidad lo mantiene el núcleo en todo momento en tiempo real.
  - Contiene un subdirectorio por cada PID de proceso activo en el sistema, con información variada sobre el mismo:
    - Ficheros abiertos.
    - Mapa de memoria del proceso, etc.
  - Contiene también información general sobre el sistema:
    - `/proc/cpuinfo`: Información sobre la(s) CPU(s) del sistema.
    - `/proc/meminfo`: Información sobre la memoria del sistema.
    - `/proc/version`: Versión del núcleo.
    - `/proc/sys`: Subárbol con parámetros de los distintos subsistemas del núcleo. Con los permisos adecuados, se puede incluso cambiar el comportamiento en tiempo de ejecución escribiendo aquí.
    - Etc.

Comunicación entre procesos

- Redireccionamiento:
  - Cada proceso tiene una entrada (*stdin*) una salida (*stdout*) y una salida de error (*stderr*) estándar.
    - Por defecto, *stdin* es el teclado, y *stdout* y *stderr* la pantalla.
  - Al lanzar un comando, se pueden redireccionar cualquiera de las salidas a un fichero (es una característica muy útil del intérprete de comandos `bash`). P.e.:
    - `ls -l > fich.txt` : En lugar de escribir la salida en pantalla (stdout), se escribe en un fichero.
    - `ls -l >> fich.txt` : Ídem, pero añadiendo al fichero al final (en lugar de "machacarlo").
    - `ls -l noexiste.c *.h 2> err.txt` : La salida de error del comando (stderr) se redirecciona al fichero err.txt. También se puede usar `2>>`, para añadir en lugar de "machacar".
    - `sort < fichero`: El comando sort (ordenar) toma la entrada de un fichero en lugar de por teclado (*stdin*).

Comunicación entre procesos

- Tuberías:
  - Siguiendo con más características útiles del `bash`, podemos "encadenar" varios comandos redireccionando el *stdout* de cada uno de ellos al *stdin* del siguiente, ensamblándolos mediante *tuberías* (`|`):
    - P.e. `cat fich.txt | sort | head -10` : Imprime las diez primeras líneas en orden alfabético de un fichero dado
  - En UNIX tradicionalmente se dispone de un repertorio de comandos muy útiles, especialmente diseñados para comunicarse entre sí.
  - A estos comandos se les denomina *filtros*, por tomar una entrada de texto en *stdin* y modificarla de alguna manera para volcar la salida filtrada en *stdout*.
  - Filosofía extraordinariamente modular y flexible.

Comunicación entre procesos

- Algunos otros comandos de interés:
  - `date`: Devuelve la fecha y la hora actuales.
  - `cal`: Un calendario del mes/año en curso.
  - `clear`: Limpia el terminal.
  - `reset`: Resetea el terminal (para cuando queda "tonto", con caracteres extraños; p.e., tras hacer `cat` de un fichero binario).
  - `expr`: Para cálculos aritméticos. P.e. `expr 101 + 12`
  - `xargs`: Muy interesante, usado con tuberías: todo lo que entra por la tubería pasa como argumento(s) del siguiente comando. Por ejemplo:
    - `find . -type f | xargs ls -l`: Lista con formato largo todos los archivos que cuelguen a cualquier profundidad del directorio actual (e.d., toda la salida del `find`, que llega por la tubería en forma de lista de caminos completos, se pasa como lista de argumentos al comando `ls -l`).
- Ahora sabemos todo lo necesario para interpretar el comando con el que empezábamos:

```
find progs -iname "*.c" -printf " + %s" | xargs expr 0
```

Índice

Arranque del sistema

- Proceso de arranque de un SO:
  - Al encender, la CPU ejecuta inmediatamente un programa en ROM (*iniciador ROM o ROM BIOS*), que:
    - Realiza un rápido autodiagnóstico del *hw* (memoria, tarjeta gráfica, discos, teclado, etc.) y comprueba que todo esté correcto.
    - Lee del disco y ejecuta el programa llamado *cargador* (que permite seleccionar, en su caso, entre varios SO presentes), y que se encarga de cargar y dejar residente el núcleo básico del SO (*kernel*) en memoria.
    - El *kernel* toma el control y establece sus estructuras internas básicas (tabla de procesos, memoria, E/S, etc.).
    - Una vez inicializado el núcleo, se lanza un proceso llamado `init`. Éste empieza a lanzar procesos auxiliares y *demonios* (impresión, red, servicios de conexión remota, etc.), según esté configurado, y finalmente, lanza uno (o varios) procesos de login, que permiten a un usuario autenticarse y empezar a trabajar.

Llamadas al sistema, Interrupciones

- Tipos de interrupciones:
  - **Interrupciones *sw***: Un proceso llama explícitamente a un servicio del sistema (llamada al sistema).
  - **Interrupciones *hw***: Algún evento reclama la atención inmediata del *núcleo* del SO:
    - P.e., provocado por un error en el proceso en ejecución (instrucción errónea, error numérico, acceso indebido, etc.; en este caso suelen denominarse más bien ***excepciones*** o ***desvíos***). Son señales generadas internamente en la C.P.U.
    - O bien se produce un evento asíncrono importante (el reloj marca final de un *quantum* del proceso, termina una operación de disco que se lanzó anteriormente, etc.). ***Interrupciones*** propiamente dichas, son señales generadas por dispositivos periféricos que ,mediante el bus de control, llegan a la C.P.U.

Llamadas al sistema, interrupciones

- En realidad, una interrupción software no es, como tal, una interrupción (no interrumpe absolutamente nada) sino que es más bien como un salto con retorno (una llamada a una subrutina ya cargada por/en la ROM-BIOS o por el S.O.) que realiza alguna tarea.
- Aunque a menudo se mantiene la terminología de "interrupciones software", heredada de Intel, en segundo cuatrimestre, en ETC, se usará la siguiente clasificación (propia de MIPS, pero más clara y correcta):

  - **Interrupciones**: Eventos externos: ejemplos típicos serían una pulsación de teclado, otra en que el reloj avisa del final de un *quantum*, u otra en la que el disco duro avisa de que acaba de finalizar una transferencia.

  - **Excepciones**: División por cero, instrucción no válida, etc.

  - **Traps**: Llamadas al sistema (syscall, o int 80h en x86).

Llamadas al sistema, interrupciones

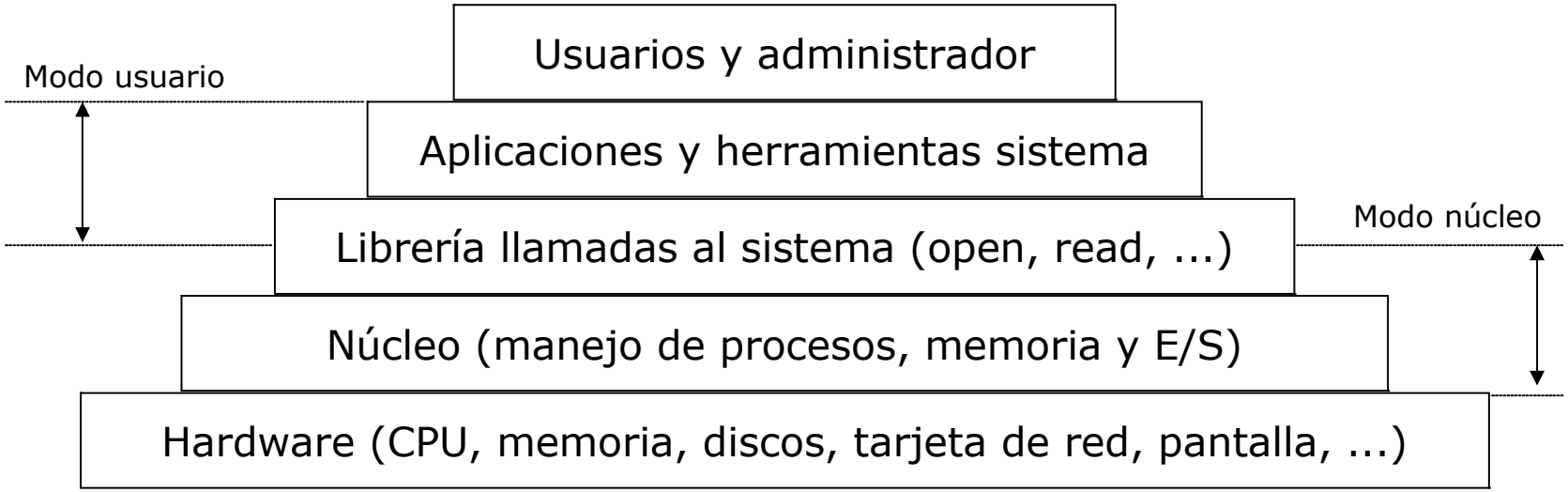
- P.e. ¿Cómo usa el programador los dispositivos de E/S?
  - Solicita el servicio al SO, mediante una *llamada al sistema*.
  - P.e. transfiere 500 bytes del fichero "fich.txt" a tal dirección de mi espacio de memoria.
  - La llamada se hace mediante una interrupción *sw* (una instrucción) que provoca que el SO tome el control.
  - El SO organiza y coordina las peticiones de los distintos procesos, comprueba permisos, y abstrae la complejidad de los distintos dispositivos:
    - Usa el manejador de dispositivo (driver) adecuado para "hablar" directamente con el periférico en cuestión
    - Red, disco, etc., tendrán manejadores muy distintos, pero el SO ofrece al programador una visión uniforme de todos ellos, para facilitar su uso (*independencia de dispositivo*)
    - Al terminar de servirse la petición, se vuelve al código de usuario en el proceso llamador.
    - Una llamada puede fallar (si no se cumple algún requisito, o se proporcionan parámetros erróneos). En ese caso, el SO maneja el fallo adecuadamente y protege al resto de procesos de una posible caída, mientras informa del fallo al proceso llamador.

Llamadas al sistema, interrupciones

- Interfaz de las llamadas al sistema:
  - Las llamadas al sistema definen el interfaz entre los programas de usuario y el SO (parámetros, valores devueltos, comportamiento deseado, etc.).
  - Existen estándares para facilitar la portabilidad entre ciertos sistemas: POSIX, librería estándar de C, etc.
- Tipos de llamadas al sistema:
  - **Procesos:** creación de nuevos procesos, terminación de procesos (ídem para hilos), etc.
  - **Acceso a dispositivos:** Apertura y cierre de ficheros, lectura y escritura (ídem para acceso a la red, teclado, pantalla, etc.).
  - **Gestión de la memoria:** Solicitud dinámica de espacio, liberación de espacio, etc.
  - **Manipulación de sistema de ficheros:** creación y borrado de directorios, movimiento por directorios, manipulación de permisos, acceso a metadatos (dueño, fecha, hora, tamaño de ficheros, ...)
  - **Otros:** sincronización y comunicación entre procesos, etc.

Visión general del sistema

- Visión por capas:



- Componentes del núcleo:

<table>
<tr><td colspan="7">Llamadas al sistema</td></tr>
<tr><td colspan="2">Manejo de terminales</td><td>Sockets</td><td>VFS</td><td>Corres–pondencia</td><td rowspan="2">Manejo de señales</td><td rowspan="2">Creación y eliminación de procesos</td></tr>
<tr><td rowspan="2">Tty cruda</td><td>Tty cocinada</td><td>Protocolos de red</td><td>Sistemas de ficheros</td><td>Memo. virtual | Fallos página</td></tr>
<tr><td>Disciplinas de línea</td><td>Enrutamiento</td><td>Caché de buffer</td><td colspan="2">Caché de páginas</td><td colspan="2">Planificador de procesos</td></tr>
<tr><td colspan="2">Manejadores de dispo–sitivos de caracteres</td><td>Manejadores de dispositivos de red</td><td colspan="2">Manejadores de dispositivos de discos</td><td colspan="2">Despachador de procesos</td></tr>
<tr><td colspan="7">Hardware</td></tr>
</table>