Sistemas Operativos

Bloque II

Informática Básica - Curso 2010/2011

Guión

1. Introducción a los Sistemas Operativos
2. El Sistema Operativo. Estructura
3. Procesos. Introducción
4. Sistemas de ficheros

Bibliografía

- Básica:
  - Carretero et al., 2007, Sistemas Operativos, una visión aplicada (2ª ed), Mc Graw Hill
- Otros:
  - Andrew S. Tanenbaum, 2009, Sistemas Operativos Modernos (3ª ed), Prentice-Hall
  - Silberschatz, A.; Galvin, P.B.; Gagne, G., 2005, Fundamentos de los Sistemas Operativos (7ª ed), Mc Graw Hill

Bloque II – Sistemas Operativos

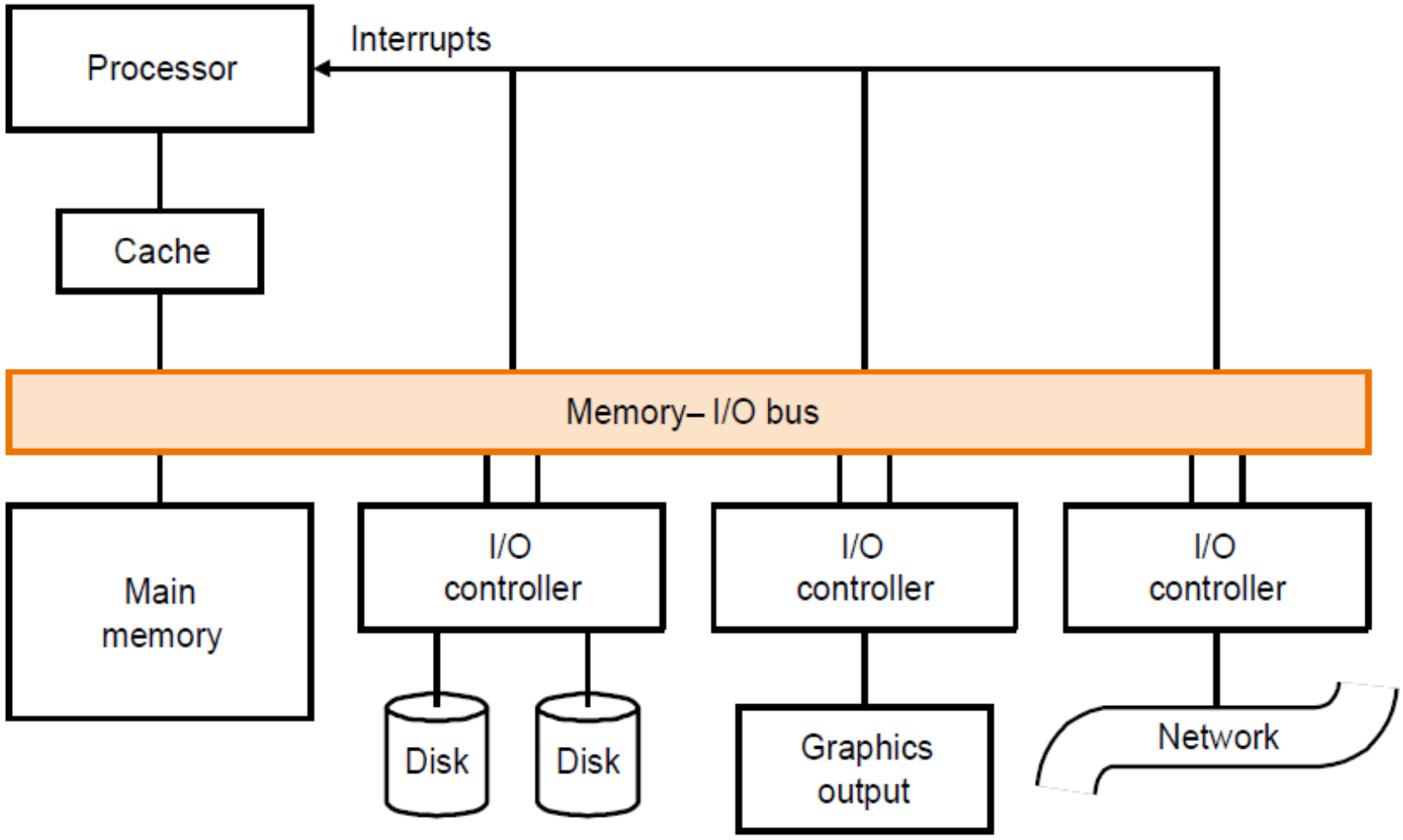
1. Introducción a los sistemas operativos

Introducción a los sistemas operativos

- Máquina desnuda
  - Ordenador carente de sistema operativo
  - Sólo el hardware, sin ningún software instalado
  - En sí mismo no hace “nada”
  - El hardware por sí mismo sólo es capaz de ejecutar programas en código máquina
    - Secuencia de instrucciones en lenguaje máquina
    - Un programador tendría que programar “TODO”!!

Arquitectura básica del computador

Fariña, Pedreira: LBD@2010



Funcionamiento del computador

- Ejecución de un programa en código máquina

1. Leer la siguiente instrucción del programa
2. Interpretar la instrucción leída
3. Leer los operandos de la memoria principal
4. Ejecutar la instrucción
5. Almacenar el resultado de la instrucción
6. Volver al paso 1

Ejemplo

- Ejemplo, muy simplificado, de programa máquina

```
…
LOAD R1, #16[R10]
LOAD R2, #32[R10]     // R2 ← Mem (R10+32)
ADD R3, R1, R2
SAVE R3, #16[R11]       //Mem (R11+16) ← R3
…
```

```
Que en pascal podría ser algo como lo
  siguiente: (aunque no sea corrento)
  res3 := var[16] + var[32];
```

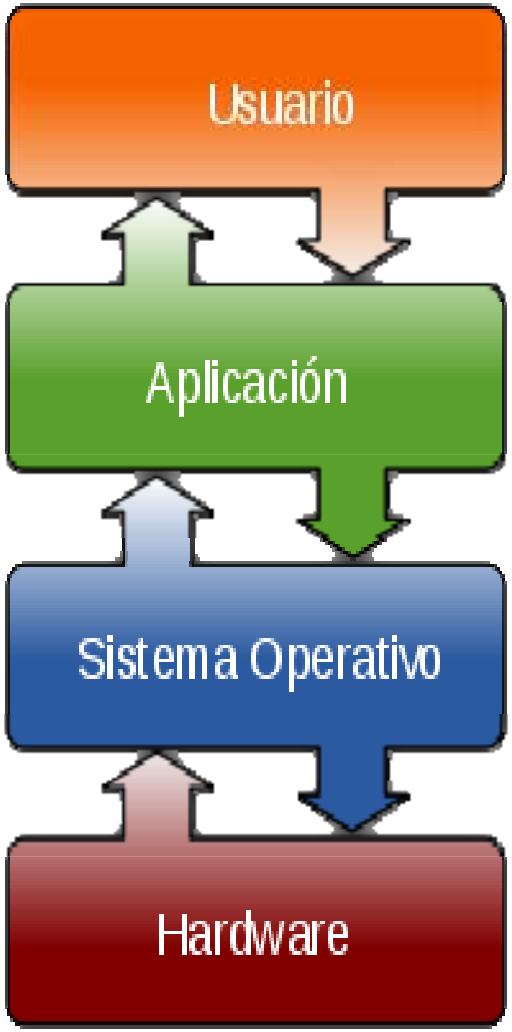
Necesidad de un sistema operativo

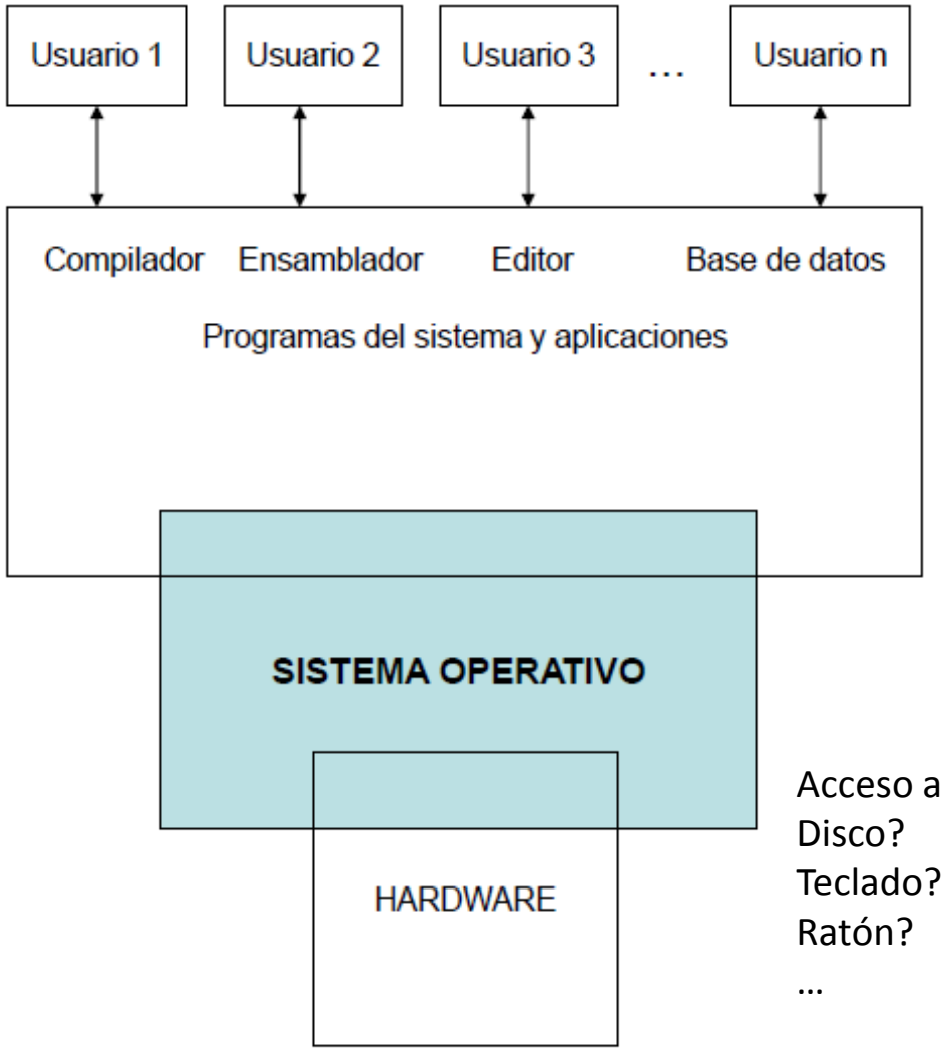
- Pero queremos algo más:
  - Ejecutar varios procesos de forma simultánea, que tienen que compartir los recursos del computador
  - Olvidarnos del detalle de uso de cada componente HW
    - P.ej: controladores de cada tipo de ratón/tarjeta gráfica,…
  - Tener un entorno de trabajo en el computador
    - Para lanzar nuestros programas, manejar ficheros (sistema ficheros),…
  - Escribir programas en lenguajes de alto nivel (compilación)
  - Tener múltiples usuarios que utilicen el sistema, posiblemente de forma concurrente.
  - Etc.

Sistema operativo

- Definición: Sistema operativo
  - Es el software de base que gestiona los recursos del computador (CPU, memoria, almacenamiento, etc.) y le proporciona al usuario un entorno de trabajo.
    - Servicios para gestión de procesos, de E/S, de gestión de almacenamiento, protección y seguridad, …
    - Asigna recursos y los recupera cuando dejan de ser utilizados.
  - Actúa como interfaz entre el hardware y el usuario y las aplicaciones que ejecuta.

Sistema operativo





Software

Acceso a :
Disco?
Teclado?
Ratón?
…

+ definiciones de S.O.,…

(1) S.O. como programa

- Podemos ver al S.O. como:
  - un conjunto de “estructuras” que almacenan información relativa a los objetos que gestiona:
    - Recursos físicos: CPU, dispositivos (disco, impresora),…
    - Recursos lógicos: Usuarios, procesos, sistemas de ficheros, etc.
  - un conjunto de “programas”, que permiten gestionar los recursos y asignar/ligar de forma eficiente los recursos físicos a los recursos lógicos. (ej: sistema ficheros → disco)
    - P.ej:

      F= fopen(“file.txt”,”w”)
      fwrite(datos, F)

      → escribe datos en bloque X de disco. (el s.f. indicará lugar)

+ definiciones de S.O.,…

(2) S.O. como máquina extendida

- Programar a bajo nivel es difícil. p.ej. a nivel de E/S
- El S.O. permite ocultar los detalles de bajo nivel al usuario y proveer un interfaz de uso común a todos los usuarios (ej. Librerías del sistema para llamadas al sistema para operaciones de E/S).
  - El programador no necesita saber cómo se almacenan los datos físicamente en un disco.
  - El controlador de disco seguirá un protocolo de comunicación con el S.O. para que el S.O. pueda almacenar datos en él.
  - A su vez, el S.O. dispondrá de llamadas al sistema que el programador podrá invocar (*open*, *read, write, close).*
    - P.ej. Ler un fichero en un S.F. NTFS o en un S.F. EXT4 es transparente al programador (aunque el S.O. sí que necesitará conocer cómo es el formato de cada s.f. info para saber cómo acceder a los diferentes tipos de s.f.)

+ definiciones de S.O.,…

(3) S.O. como administrador de recursos

- El SO ha de llevar *contabilidad* del uso de los recursos disponibles.
  - Los programas usan los recursos existentes (cpu, memoria, espacio en disco, una impresora, un fichero abierto)
  - El S.O. debe saber qué recursos están siendo usados en cada momento.
  - El S.O. se encargará de la asignación de recursos.
  - El S.O. debe encargarse de recuperar (establecerlos como disponibles) los recursos que hayan dejado de estar en uso

Sistema Operativo: pdv gestión

- Desde el pdv del **entorno de ejecución de Programas**
  - Permite cargar programas
  - Gestiona las operaciones de E/S
  - Permite la manipulación del Sistema de Ficheros
  - Detecta errores
  - …
- Desde el pdv de la **eficiencia**, los servicios del SO. son:
  - Asignación de recursos
  - Contabilidad
  - Protección
  - …

Sistema operativo

- El S.O incluye:
  - Núcleo o *kernel*. Constituye el elemento central del S.O.
    - Instrucciones básicas para:
      - Arranque del computador
      - Inicio de servicios básicos
      - Interfaz con el hardware
    - Otras aplicaciones más enfocadas al usuario (entorno)
      - Intérprete de comandos
      - Apps de usuario (gestión red, usuarios,…)

Sistema operativo: ejemplos actuales

- Familia Unix: Partieron de una raíz común y siguieron caminos separados.
  - Linux (Debian, Ubuntu, Suse, Red Hat,…)
  - Solaris (Sun)
  - FreeBSD,…
- Windows: (Ms)
  - DOS + Windows 3.1x
  - Windows 95, 98, Me, 2000, Xp, Vista, 7
  - NT, 2000server, 2003server, 2008
- Mac Os (Apple)
- Android (Google para móviles)

Sistemas Operativos actuales: características

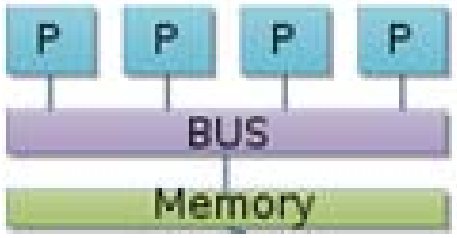
- Características técnicas
- Capacidad/facilidad de administración
- Robustez y soporte
- Consumo de recursos
- Facilidad de uso
- Coste

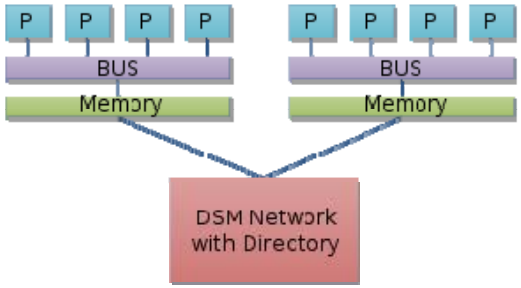
Sistemas Operativos: Tipos (complejidad)

- Nº usuarios:
  - Monousuario
  - Multiusuario
- Procesos simultáneos que puede ejecutar:
  - Monotarea: un único proceso. No hay concurrencia
  - Multitarea (o tiempo compartido).
    - varias tareas simultáneas.
- ¿S.O. Multitarea sobre máquina "*single core*"?
  - Tiempo compartido (quantum)
  - Concepto de planificación

Sistemas Operativos: Tipos (complejidad)

- S.O. para arquitecturas multiprocesador. (*dual-core, quad-core, N-core)*
  - 2 procesos o + *no simplemente alternan.*
    - Se ejecutan “en paralelo” y pueden querer acceder a los mismos recursos. En concreto, a la memoria.
  - Varias arquitecturas (SMP, NUMA, ccNUMA,…)
    - SMP (Symmetric multiprocessing)… mem compartida
    - NUMA (*Non-uniform memory access).* Cada cpu accede a una parte “exclusiva” de la memoria. El acceso no es igual de rápido a cualquier región de memoria





Sistemas Operativos: Tipos (complejidad)

- S.O. en red/distribuido.
  - No centrado en el gobierno de una máquina!
  - sino en el de múltiples computadoras (hasta miles!) conectadas a través de una red.
    - Escalabilidad
  - No comparten espacio de memoria
    - Protocolo de paso de mensajes.

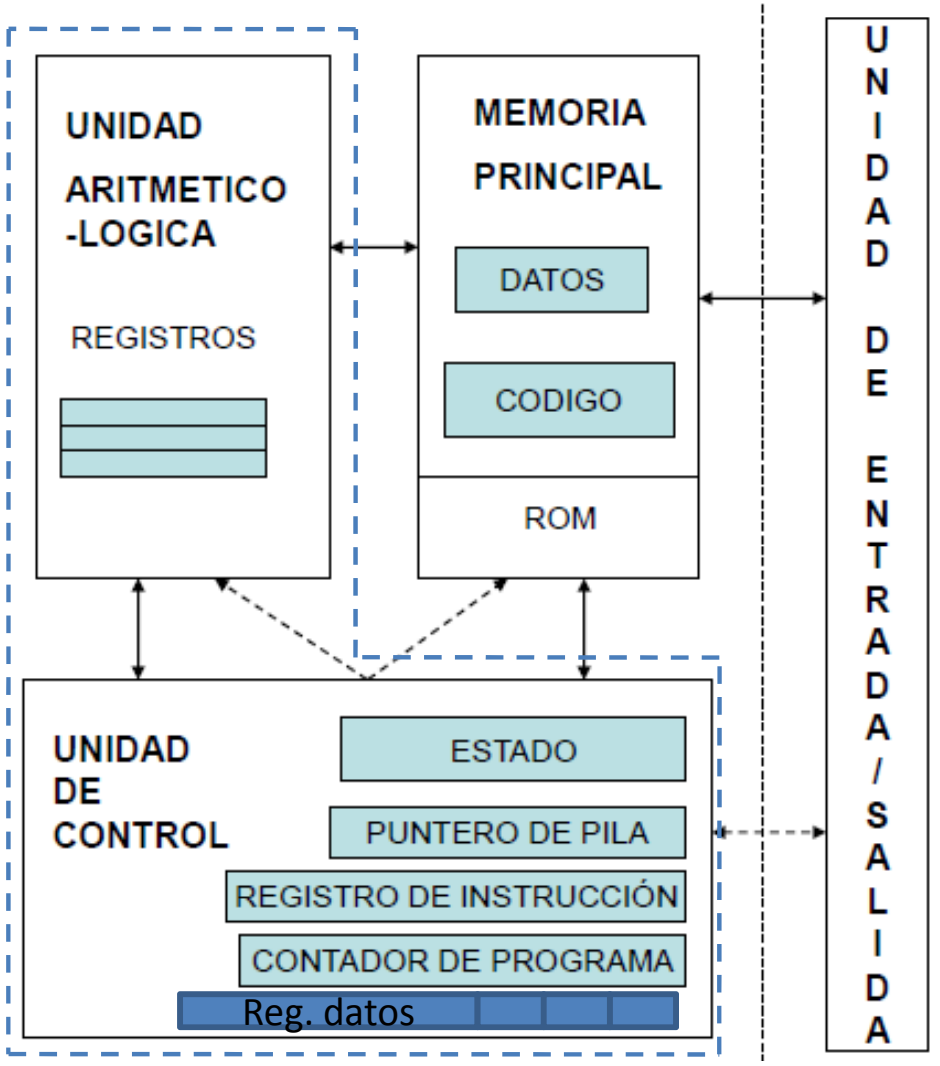
Sistemas Operativos: Más tipos (uso)

- **Sistema modo interactivo** (el usuario "dialoga" con los procesos a través de un terminal) VS **sistema modo batch** o por lotes (se usa una cola de trabajos que son atendidos cuando el sistema tiene tiempo).
- S.O. de *escritorio*. Para ordenadores personales.
- S.O. de *servidor*. Dar servicios a un conjunto de usuarios.
  - Con + servicios para **monitorización** y **admin.** del sistema.
- S.O. *para dispositivos móviles*.
  - Hardware con menos recursos, interfaz gráfico más pobre?
  - S.O. menos servicios disponibles, requiere menos recursos.
  - P.ej. Palm OS, Windows CE/mobile, Symbian OS.
- S.O. sistemas *embebidos/empotrados* (en un sistema + grande)
  - P.ej. S.O. centralita de un coche.
- S.O. de *propósito específico* (para una máquina en concreto)
  - P.ej. Una videoconsola.
- S.O. de *tiempo real*. Deben garantizar que los procesos se ejecuten en un tiempo dado para reaccionar a las necesidades del sistema
  - P.ej. Sistema de Ctrl de una central eléctrica.

Estructura y funcionamiento computador

Fariña, Pedreira: LBD@2010

- **A.L.U.**
  - Operaciones aritméticas y lógicas
- **U. Control**
  - Lee de mem sig instruc. máquina
  - La interpreta
  - Lee de memoria datos si necesario
  - Ejecuta instrucción
  - Almacena resultado (si hay)
- **R.E.** Contiene info de usuario: sobre la ejecución de las últimas instrucciones (overflow,zero, acarreo..).
  Info del sistema: Máscara interrupción, modo ejecución.
- **PC.** Contador prog. Dir sig. Instrucción.
- **RI.** Almacena la instrucc a ejecutar (leída de memoria previamente)
- **SP.** Stack Pointer. Necesaria para saltos en nuestro programa (guardar estado, dirección de retorno,…)

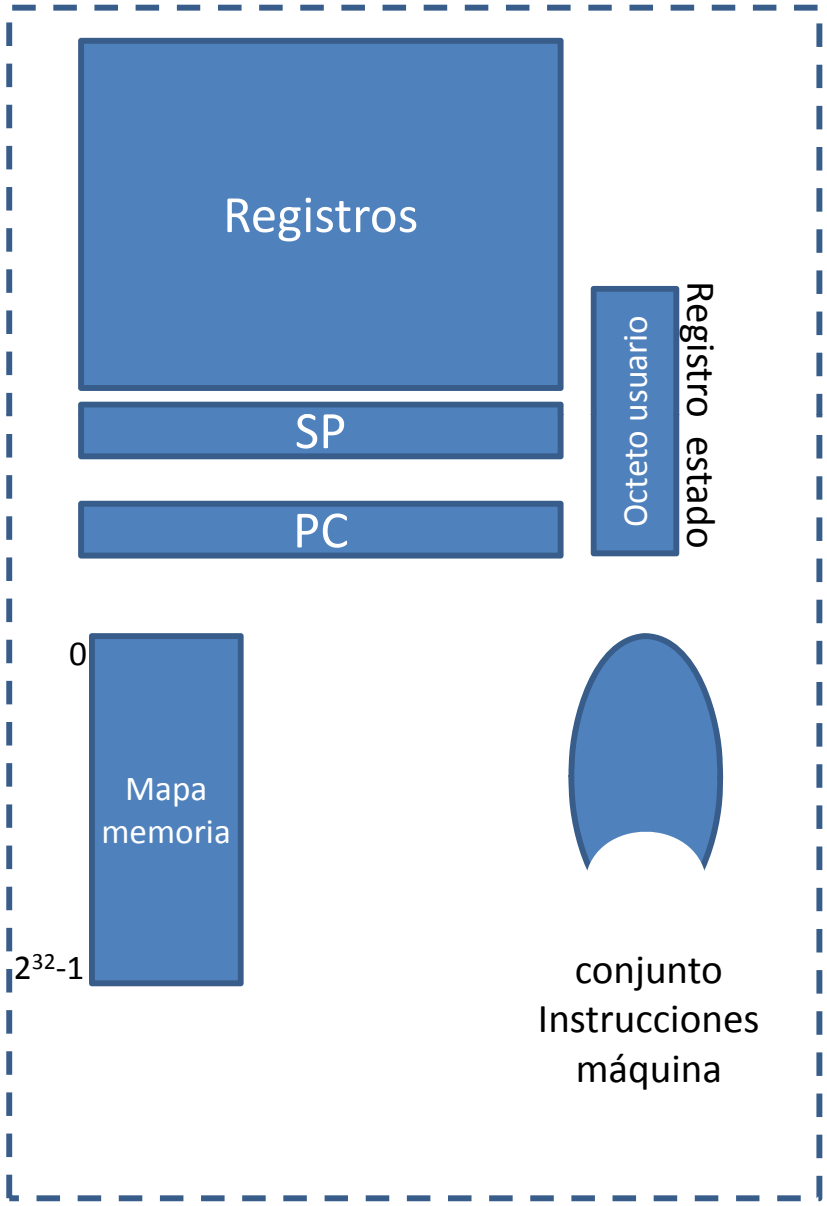


CPU: UC + ALU

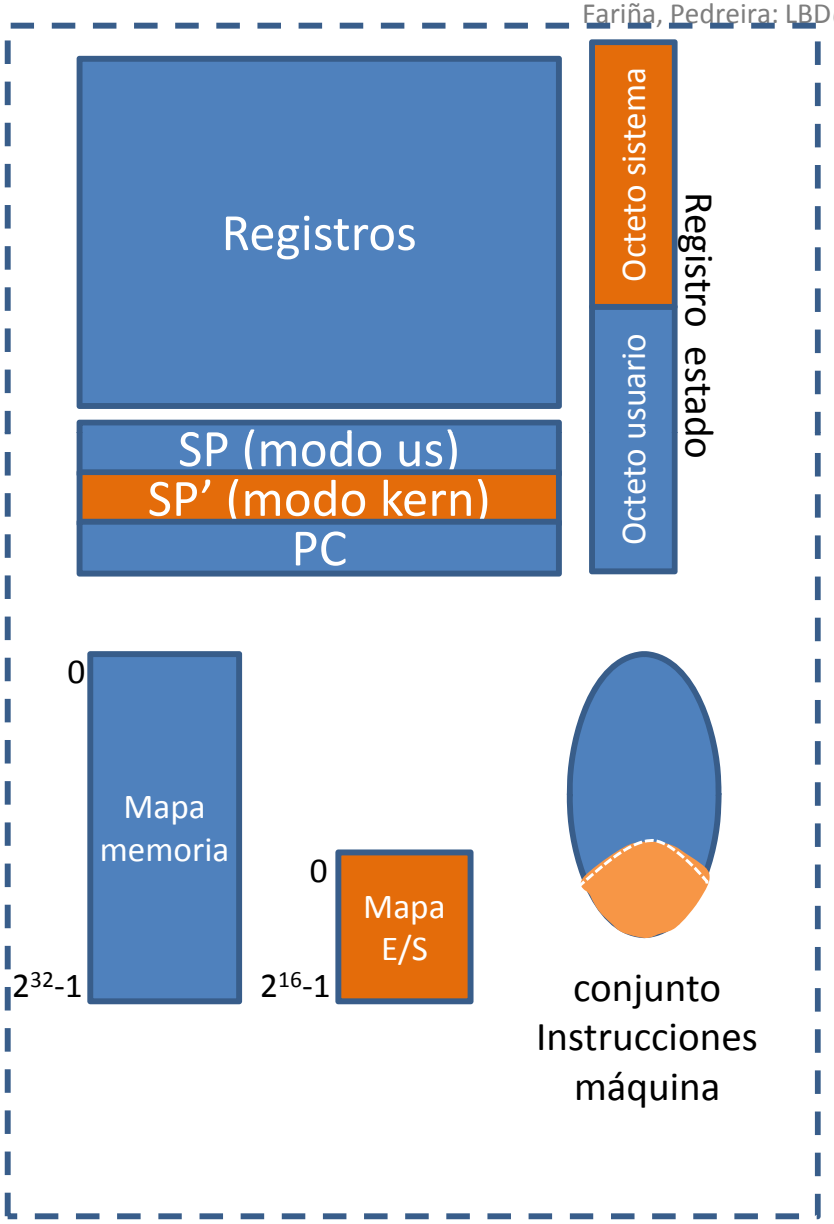
CPUs: 2 Modos de ejecución (o más)

- 2 modos de ejecución
  - Modo *kernel*
    - La CPU permite acceso a todos sus objetos
    - Instrucciones privilegiadas
    - El SO (programa) se ejecuta en este modo.
  - Modo usuario
    - Las instrucciones disponibles de la CPU son menos
    - Permiten acceder a menos objetos
      - p.ej.
        - » El mapa de E/S. no está disponible (info registros controladores)
        - » Acceso a memoria protegida no es permitido.
        - » El juego de instrucciones máquina usables es más reducido.
    - Los programas de usuario se ejecutan en este modo.
  - Estos 2 modos, facilitan la implantación de seguridad en nuestro sistema.
    - Ej. un usuario no podrá acceder a la memoria de otro.
- Instrucción TRAP (cambio de modo). Y llamadas al sistema.
  - Método que posibilita realizar acciones no permitidas en modo de usuario (se invoca al sistema). Ej: abrir un fichero, solicitar memoria al SO,…

CPUs: 2 Modos de ejecución (o más)



Modelo de programación: modo usuario



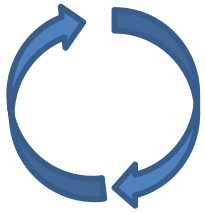
Modelo de programación: modo privilegiado

Estado del procesador y estado visible

- Estado del procesador
  - Lo constituye el conjunto de todos los registros de la CPU
- Estado visible
  - Parte del estado del procesador que es *visible* en modo usuario

Funcionamiento procesador

- Repite a gran velocidad:
  - Cargar instrucción apuntada por el **PC**.
  - Incrementar PC.
  - Ejecutar instrucción



- Ejecución lineal del programa es muy limitada
  - Guardar *estado* y … modificar al *dirección del PC*…
  - Existen mecanismos para "**saltar**": a otra región del programa, o a otro programa.
    - Instrucciones de salto o bifurcación (if, llamada función,…)
    - Interrupciones internas o externas (salto a otro programa [RTI], que es parte del S.O.)
    - Instrucción máquina de "llamada al sistema" (TRAP, INT,…). Es la forma de "acceder al S.O." para requerir sus servicios (p.ej: *read, write,…*)

Interrupciones

- Se activan con una señal que llega a la U.Control (UC)
- Al llegar la señal, si esta interrupción está activada, la UC inicia un ciclo de activación de la interrupción.
  - Salva algunos registros del procesador (en la pila SP')… entre ellos el valor actual del PC.
  - Eleva el modo de ejecución a modo *kernel*.
  - Carga en el PC un nuevo valor: la dirección de la rutina de tratamiento de la interrupción (RTI)… que pasa a ejecutarse.
    - La dirección base de la RTI está ***protegida***, para que en modo usuario no puedan ejecutarse!
  - Dependiendo de cómo se gestionen la interrupciones, puede inhabilitar temporalmente las interrupciones (para que la rutina de tratamiento de la interrupción no se pueda ver interrumpida por otra interrupción).

REGRESO DE LA INTERRUPCIÓN:

Cuando finaliza la RTI, se restablecen los datos guardados en la pila (SP') y el PC, para continuar con la ejecución normal del programa anterior. Este retorno de la RTI se realiza mediante una instrucción de "retorno de la RTI". Dicha instrucción de conoce como **RETI**.

Interrupciones: causas que las generan

Condiciones anómalas

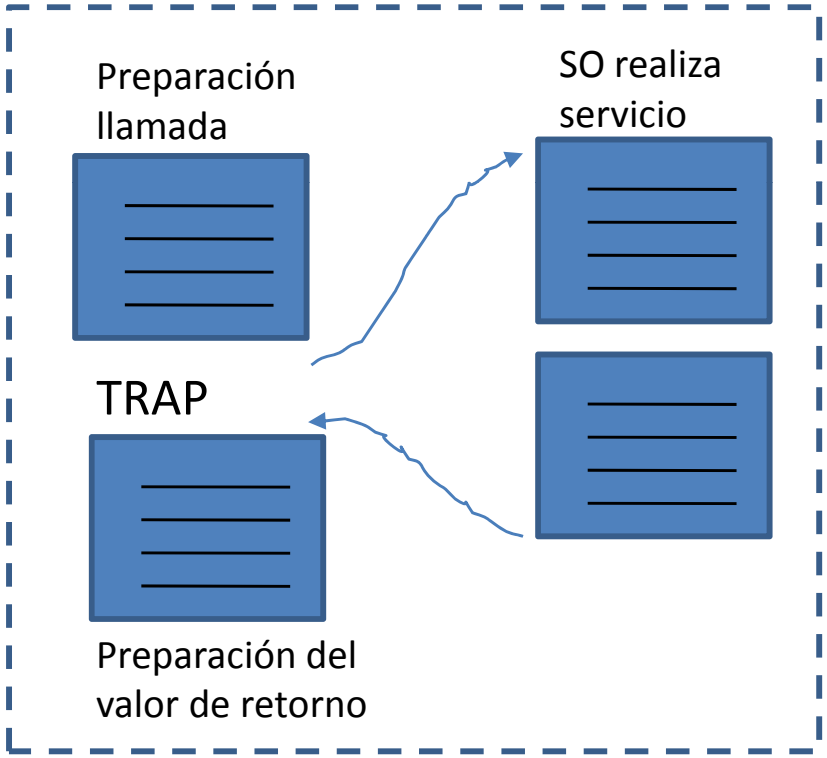
Condiciones normales

- Excepciones hardware síncronas (o excep. Software)
  - Problemas de ejecución (de la instrucción ejecutada)
    - División por 0.
    - Desbordamiento en resultado 3.000.000.000 + 3.000.000.000
    - Desbordamiento Pila (*stack overflow*)
    - Dispositivo no existente (p.ej. Coprocesador)
    - …
  - Depuración (debuggers). *Breakpoints* en el depurador.
  - Fallo de página (debe cargarse una nueva página de mem)
- Excepciones hardware asíncronas
  - Error de paridad (en bus, en memoria),
  - …
- Interrupciones externas
  - Reloj, controladores de dispositivos de E/S, otros procesadores.
- Instrucciones máquina de llamadas al sistema (TRAP, INT)…

Acceso a los servicios del S.O.
Llamadas al sistema

- Se realizará a través de funciones de biblioteca que nosotros usaremos en nuestros programas.

- Partes función biblioteca:
  - Preparación del código y de los argumentos de la llamada
  - Ejecución de la instrucción **TRAP**, para pasar a ejecutar el S.O. (rutina que realiza el servicio)
  - Recuperación de los resultados devueltos por el S.O. y preparación del valor retornado
- Ejemplos: n=fork(), l=read(),…



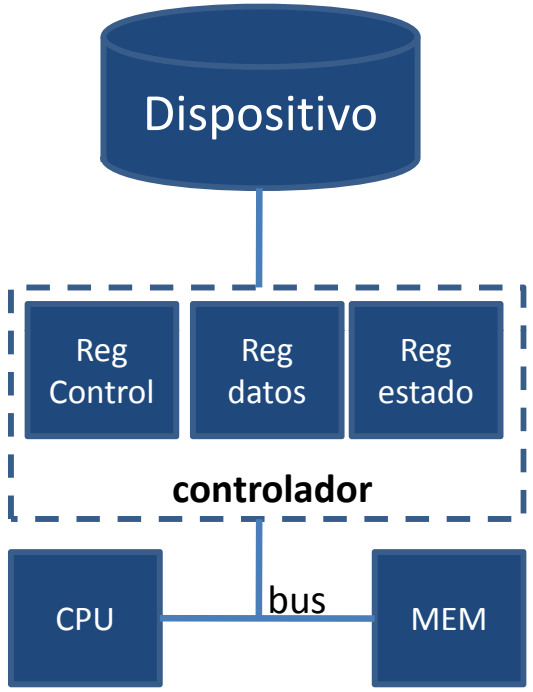
Rutina de biblioteca para llamar a un servicio del S.O

El reloj: Protección de la CPU y +

- Tres significados diferentes del *reloj*
  - CLK (ciclo de reloj) 1GHz → ciclo reloj = 1ns
  - Temporizador o generador de interrupciones periódicas
    - Relacionado con la *protección de la CPU*.
    - Hace que el SO se pase a ejecutar periódicamente (*tick*)
      - 1 programa de usuario no se puede "apropiar" de la CPU!!
  - *Real Time* Clock (RTC)
    - Fecha y hora actuales (en UNIX, cuenta los segundos que han pasado desde 1/1/1970 (número entero de 32bit)

Entrada/Salida

- E/S. Intercambio de información entre:
  - los periféricos y
  - la memoria o los registros de la CPU
- Controlador:
  - Registro de Datos:
    - Lugar dónde el controlador irá cargando los datos (leídos) o poniendo los datos que se enviarán al periférico.
  - Registro de Estado:
    - Indica si los datos (leídos) o pendientes de escritura (para escribir), ya han sido colocados en el Reg. Datos. → El controlador puede "transferir la palabra".
  - Registro de Control:
    - Le indica al controlador la operación a realizar.



Estructura general e/s.
Periférico + controlador
Mem + CPU

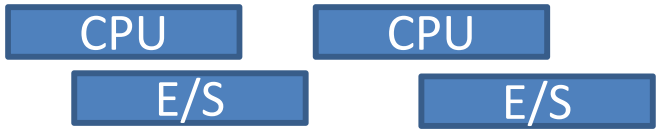
Formas de E/S: concurrencia de E/S

- E/S programada
  - Obliga a la CPU a estar ejecutando un programa de E/S.

    (ESPERA ACTIVA: un bucle que lea el estado del controlador y mire en el registro de estado si hay datos disponibles)
  - La CPU debe esperar (y chequear) a que el dispositivo actualice su estado
  - No hay ningún tipo de concurrencia entre CPU y E/S
    - Mientras el controlador “lee/escribe” datos, la CPU no procesa nuevo trabajo.

Formas de E/S: concurrencia de E/S

- E/S con interrupciones o por DMA
  - La CPU no espera mientras la E/S se completa y puede estar ejecutando otro programa
  - **3 fases**:
    - Envío de la orden/ lectura o escritura datos / fin de operación



  - **E/S con interrupciones:** tras envío de orden, la CPU puede ejecutar otro programa.
    - Cuando el controlador haya "leído/escrito" **un dato** generará una interrupción externa.
    - En la rutina de manejo de la interrupción, la CPU carga el dato (si lectura) en memoria.
  - **E/S con DMA**.
    - El controlador carga los datos a memoria de forma autónoma (se indica la posición durante la primera fase de *envío de orden*)
    - Sólo envía interrupción cuando haya leído **todos los datos**

Ejemplo E/S

- PERIFÉRICO LENTO:
- **Impresora**: 20 págs /minuto (1 pág = 3.000 chars) → 60.000 bytes /minuto
- CPU a 200Mhz (1 ciclo = 5ns). Asumiendo ciclo medio por instrucción CPI=2ciclos.
  - 1 instrucción tarda en promedio 10ns en ejecutarse → CPU ejecuta hasta 100MIPS.
- E/S programada:
  - La cpu envía un byte cuando la impresora está lista para recibirlo (y espera)
  - Impresora tarda 1sg en recibir 1000bytes
  - Por cada 1000 bytes enviados a la impresora, **la CPU espera 1sg**→ en ese tiempo podría haber ejecutado 100M de instrucciones!
- E/S con interrupciones:
  - Impresora genera interrupción cuando esté preparada para recibir 1 nuevo byte
  - Suponiendo que la RTI consume 10 instrucciones (salvar contexto, comprobar estado, transferir byte, restaurar contexto, RETI)
  - Para transferir 1000bytes ejecuta 1000 RTI → Hay que ejecutar 10.000 instrucciones para atender al periférico → la cpu está ocupada **0.0001sg** para transferir 1000 bytes
- La E/S con interrupciones es 10.000 veces más rápida que la E/S programada para atender a nuestra impresora.

Ejemplo E/S: Necesidad de DMA

- PERIFÉRICO RÁPIDO:
- **Disco usb:** Transferencia 10M bytes/s (1 byte cada 2*10^7 sg)
- CPU a 200Mhz (1 ciclo = 5ns). Asumiendo ciclo medio por instrucción CPI=2ciclos.
  - 1 instrucción tarda en promedio 10ns en ejecutarse → CPU ejecuta hasta 100MIPS.
- Transferir 10Mbytes de memoria a disco-usb
- E/S programada:
  - La cpu envía un byte (y espera) cuando HDD está listo para recibirlo
  - HDD tarda 1sg en recibir 10M bytes.
  - La **CPU espera 1sg**→ en ese tiempo podría haber ejecutado 100M de instrucciones!
- E/S con interrupciones:
  - HDD genera interrupción cuando preparada para recibir 1 nuevo byte (EXAGERADO!)
  - Suponiendo que la RTI consume 10 instrucciones (salvar contexto, comprobar estado, transferir byte, restaurar contexto, RETI)
  - Para transferir 10M bytes se ejecutan 10MillonesRTI

- ¿Es mejor E/S programada o E/S por interrupciones? ???

Ejemplo E/S: Necesidad de DMA

- PERIFÉRICO RÁPIDO:
- **Disco usb:** Transferencia 10M bytes/s (1 byte cada 2*10^7 sg)
- CPU a 200Mhz (1 ciclo = 5ns). Asumiendo ciclo medio por instrucción CPI=2ciclos.
  - 1 instrucción tarda en promedio 10ns en ejecutarse → CPU ejecuta hasta 100MIPS.
- Transferir 10Mbytes de memoria a disco-usb
- E/S programada:
  - La cpu envía un byte (y espera) cuando HDD está listo para recibirlo
  - HDD tarda 1sg en recibir 10M bytes.
  - La **CPU espera 1sg**→ en ese tiempo podría haber ejecutado 100M de instrucciones!
- E/S con interrupciones:
  - HDD genera interrupción cuando preparada para recibir 1 nuevo byte (EXAGERADO!)
  - Suponiendo que la RTI consume 10 instrucciones (salvar contexto, comprobar estado, transferir byte, restaurar contexto, RETI)
  - Para transferir 10M bytes se ejecutan 10MillonesRTI → Hay que ejecutar 100M instrucciones para atender al periférico → la cpu está ocupada **1sg** para transferir 10M bytes
- La E/S con interrupciones no mejora el tiempo en este caso.

Ejemplo E/S: Necesidad de DMA

- En periféricos rápidos interesa DMA.
  - Evita un número elevado llamadas a las RTI (Rutina Tratamiento de Interrupción).
  - CPU participa en la fase de iniciación de los parámetros de transferencia
  - CPU no interviene en la fase de transferencia de Datos
  - Cuando recibe interrupción de “transferencia finalizada” los datos ya están en memoria.

Arranque del Sistema

Fases principales:

- Iniciador ROM.
  - Realiza el test de hardware (detectar componentes: memoria, cpu, hdds, periféricos,…)
  - Carga en memoria el cargador (boot) del S.O. que estará almacenado en una parte prefijada (normalmente el primer bloque de disco). Para ello el iniciador ROM utiliza rutinas básicas almacenadas en la *BIOS (Basic Input-Output System)* que permiten acceso a disco, a teclado, acceso a la pantalla,…
- Cargador del S.O. (boot)
  - Se encarga de traer a memoria y ejecutar el programa de arranque del S.O. Incluye las operaciones siguientes:
    - Comprobar el sistema de ficheros
    - Se carga en RAM la parte *residente* del SO (que ha de estar siempre en memoria)
    - Se establecen estructuras de información propias del S.O. (tabla de interrupciones, tabla de procesos, tablas de memoria,…)
    - Se activa la gestión de memoria virtual (MMU)
    - Se habilitan las interrupciones
    - Se crea el proceso de inicio o login por cada terminal del sistema, y procesos auxiliares y demonios del sistema (impresión, red, …)
- Fase de funcionamiento normal del S.O.

Parada del sistema

- El S.O. mantiene en memoria RAM gran cantidad de información crítica: entre ellos, parte de los ficheros que se están utilizando (*buffer caché y otros datos!*).
- **Parada ordenada**: Debemos dar tiempo a que copie (actualice) a disco todos los datos críticos que estaban en memoria y termine todos los procesos activos.
  - P.ej.: Similitud: “retirar hardware USB con seguridad”
- Si parada brusca, el S.O. ha de chequear consistencia de los datos de disco (y esperar a que todo vaya OK!)
- **Apagar/hibernar/suspender.**
  - Apagado normal. Se para el computador y se interrumpe la alimentación.
  - Hibernar: no se hace apagado, se vuelca a disco la info de la memoria principal. Se mantendrá estado programas. Se interrumpe la alimentación.
  - Suspensión: Se para todo el computador, pero se mantiene la alimentación de la RAM para que pueda mantener los datos.

Bloque II – Sistemas Operativos

2. El Sistema Operativo. Estructura

Componentes del S.O.

- Gestión de procesos
- Gestión de memoria
- Comunicación y sincronización entre procesos
- Gestión de E/S
- Gestión de almacenamiento secundario: ficheros y directorios
- Seguridad y protección

- !! Los componentes pueden ofrecer sus servicios a través de una interfaz (comando/función de biblioteca) que permite invocar algún servicio del sistema (p.ej: realizar llamadas al sistema)!!

Gestión procesos

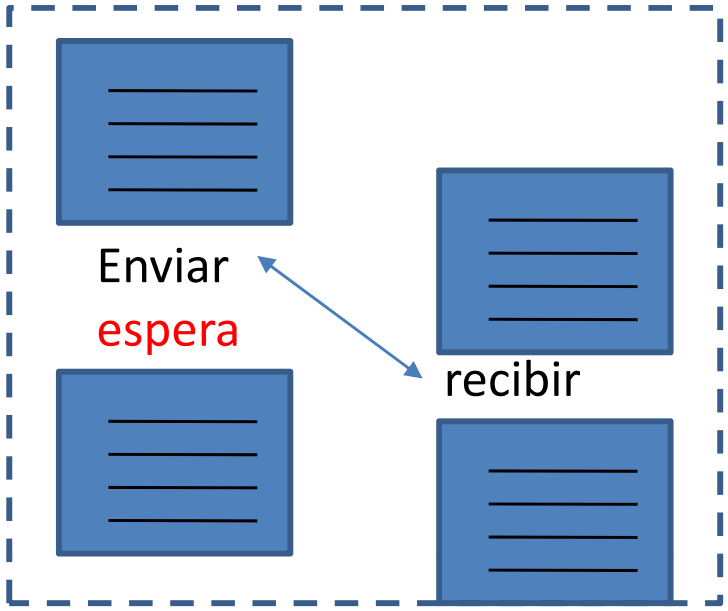
- Un proceso se puede ver como un programa en ejecución
- Tarea primordial del S.O.→ ejecución programas.
- Procesos necesitan recursos:
  - Memoria: El programa (instrucciones) y sus datos han de residir en memoria principal.
- El S.O. debe conocer el estado (reg. CPU e imagen de memoria asignada)
- Servicios:
  - Crear y eliminar procesos del sistema (fork, kill)
  - Ejecutar, suspender, reanudar procesos. (sleep, kill CONT)
  - Cambiar ejecutable de un proceso (execv)
- Planificación de procesos, gestión prioridad procesos…

Gestión de memoria

- SO ha de asignar memoria a procesos para crear su imagen en memoria
- Asignar memoria solicitada y liberarla cuando los procesos lo soliciten (control memoria asignada/libre)
- Tratar los errores de acceso a memoria (protección: un proceso no puede interferir en la ejecución de otro)
- Comunicación y compartición memoria entre procs.
  - P.ex. Segmentos de memoria compartida IPC.
- Gestión de la jerarquía de mem y tratamiento de los fallos de página
- Servicios:
  - Solicitar memoria (malloc)
  - Liberar memoria (free)
  - Compartir memoria (shmget,…). Gran facilidad comunicación.

Comunicación y sincronización procesos

- Procesos pueden colaborar en la realización de una tarea. Para ello, deben:
  - poder comunicarse entre sí para transmitir datos y órdenes.
  - Sincronizarse en la ejecución de sus acciones.
- Servicios comunicación
  - Crear tuberías/pipes, memoria compartida, sockets
  - Enviar o recibir datos (write/read)
- Servicios sincronización
  - Semáforos
    - Creación (sem_get, sem_init)
    - Esperar (sem_wait)… bloquear proceso
    - Despertar (sem_post)
    - Destrucción (sem_destroy)
  - Nota: Problemas clásicos sincronización



Comunicación síncrona entre procesos

Gestión de E/S

- El S.O. lleva la gestión de los periféricos. Objetivos:
  - Facilitar su manejo. Interfaz genérica, sencilla, uniforme y fácil de utilizar de acceso, y que debe gestionar los **errores** que puedan surgir del uso de un periférico.
  - Facilitar acceso eficiente dispositivos de almacenamiento: p.ej. buffer caché.
  - Garantizar protección de E/S. Evitan el acceso no autorizado a dispositivos periféricos a los usuarios.
  - EL S.O. ha de gestionar: relojes, terminales, dispositivos de almacenamiento 2ario y 3ario.,…
- Servicios:
  - orientados a la lectura/escritura de datos en el periférico.
  - Independientes del tipo de dispositivo.
    - Ej. Read/write → escribir leer datos en un floppy o hdd.

Gestión almacen. 2ario: Ficheros/directorios

- El gestor de ficheros ha de:
  - Facilitar el acceso a dispositivos periféricos (y de almacenamiento).
    - Visión lógica simplificada en forma de ficheros y *ficheros especiales* (dispositivos: terminales [tty], impresoras [lp],…)
    - Protección usuarios. Limitaciones a los ficheros que puede manejar cada usuario.
  - **Visión lógica**: Objetos (ficheros y directorios) organizados por un nombre lógico diferente.
    - Ejemplo: un fichero tiene un nombre que lo identifica, y es visto como una secuencia de bytes, y un puntero de posición “actual”.
  - **Visión física**: Detalles sobre cómo están proyectados dichos objetos sobre los periféricos correspondientes (p.ej. Discos → ntfs, ext4,…).
    - Ejemplo: de un fichero me importa cómo está organizado en disco (bloques, y cómo se accede a: (1) sus bloques de datos, y (2) la información “física” que almacena el s.f. sobre él como puede ser el tamaño, permisos de usuario, fecha de creación, etc.
      - Esta información es transparente para el usuario del sistema

Gestión almacen. 2ario: Ficheros/directorios

- **Servicios de ficheros:** dirigidos al manejo de datos
  - Abrir, cerrar, crear, borrar, leer, escribir, posicionar, stat.
    - Nombre fichero para "abrir y crear" → descriptor tabla ficheros abiertos
    - Resto servicios usan dicho descriptor para referirse al fichero ya abierto.
- **Gestión de directorios**: relacionan de forma unívoca un nombre con un fichero. No puede haber 2 ficheros que tengan el mismo nombre (ruta).
  - Visión lógica: organización en forma de árbol.
    - El S.O. mantiene el "directorio actual"
    - ruta absoluta y relativa para identificar a los ficheros (cq. tipo de fichero)
  - Visión física: Cada directorio almacena sus ficheros y subdirectorios en una tabla que está almacenada "en un bloque (o +) de disco".

    Dicha tabla relaciona: "nombre entrada" ←→ "identificador físico" lo que permite, para cada entrada, localizar su descripción "física"

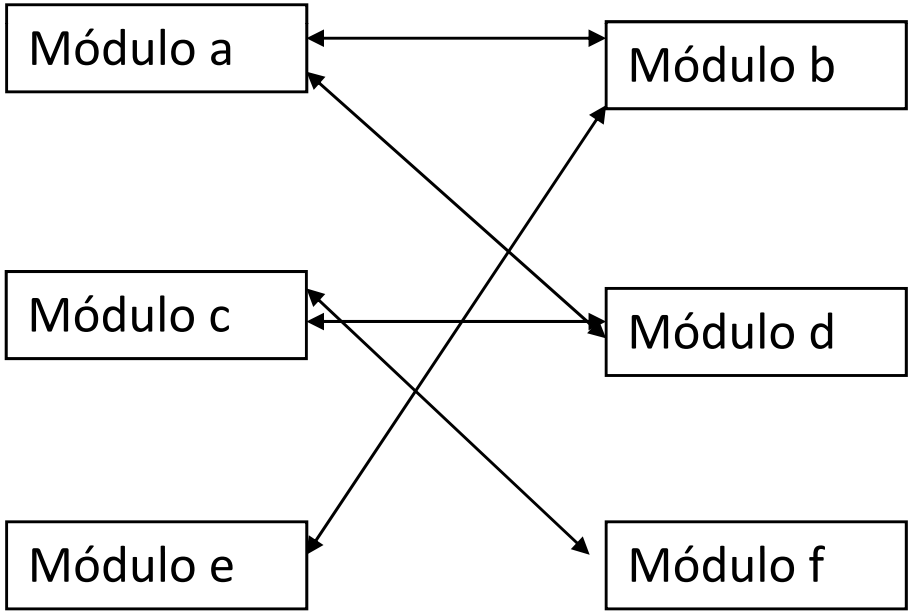
    (p ej: nombre -> inode)
  - **Servicios de directorios**:
    - Abrir, cerrar, crear, borrar, leer directorio, dir actual, cambiar directorio (LINUX: *opendir, closedir, mkdir, rmdir, readdir, cwd, chdir*)

Gestión seguridad y protección

- Seguridad
  - Evitar la pérdida de “bienes” (datos o equipamiento)
  - Controlar su uso (privacidad datos, utilización equipamiento).
- Protección
  - Evitar el uso indebido de recursos del computador
    - Memoria: un proceso sólo puede usar su espacio de direcciones. (garantizado por el hw de direccionamiento de memoria)
    - CPU (el reloj evita que un proceso se apropie de la cpu)
    - E/S. Un usuario no puede “por su cuenta” realizar E/S. Así protegemos la integridad de los disp. Físicos.
  - Aspectos a considerar:
    - Autenticación (login/pass). Verificar identidad de 1 usuario.
    - Privilegios. Operaciones que cada usuario puede realizar sobre un recurso dado.
      - El SO. Debe apoyarse en mecanismos hardware que le permitan vigilar:
        - » para cada operación, si está permitida o no,
        - » para cada acceso a memoria, si dicho acceso está permitido o no.

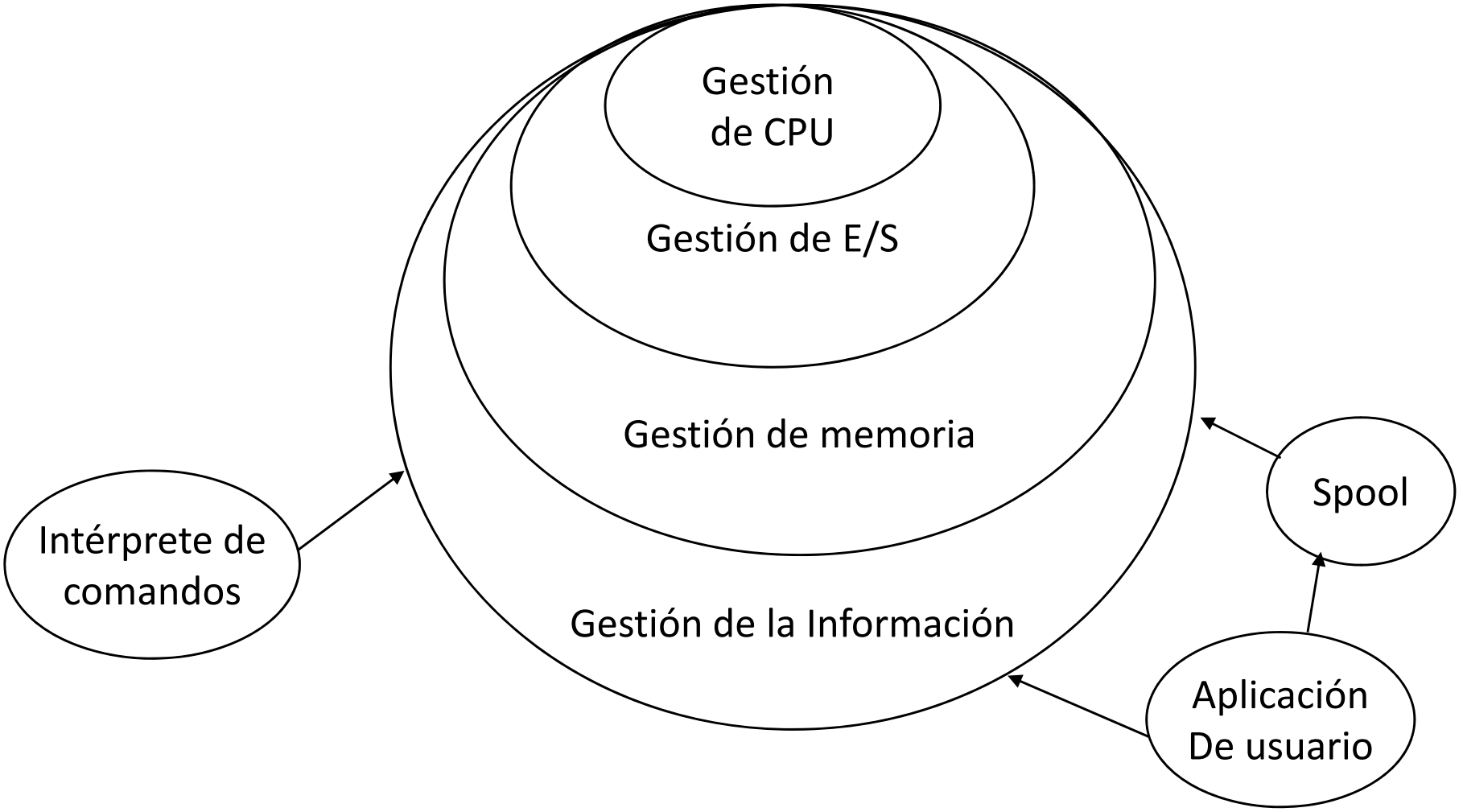
Componentes del S.O.

- Estructura monolítica del S.O.
  - El S.O. es un “todo”



Componentes del S.O.

- Estructura jerárquica (en capas) del S.O.



Servicios: Pero además…

- Un S.O. no es actualmente sólo una herramienta que sólo gestiona los recursos del sistema
  - Proporciona servicios que hace más cómoda la labor del usuario del sistema.
  - Utilidades que, si bien no pueden considerarse SO., se consideran utilidades imprescindibles en todo sistema informático y suelen venir a disposición del usuario con el S.O.. Estas podemos dividirlas en:
    - **Manipulación de archivos**. Estos programas crean, eliminan, copian, renombran, imprimen, vuelcan, muestran y en general manipulan archivos y directorios.
    - **Información de estado**. Algunos programas simplemente solicitan al sistema fecha, hora, espacio de memoria o disco disponible, número de usuarios... etc.
    - **Modificación de archivos**. Es normal que exista algún editor que permita la manipulación de ficheros en sus aspectos básicos.
    - **Apoyo a lenguajes de programación**. Es usual que se ofrezcan compiladores para los lenguajes más habituales. Hoy en día se ofrecen por separado.
    - **Carga y ejecución de programas**. Una vez compilado el programa, se debe cargar en la memoria para ejecutarlo.
    - **Comunicaciones**. Estos programas proporcionan el mecanismo para crear conexiones virtuales entre procesos, usuarios y diferentes sistemas de computación; permiten a los usuarios enviar mensajes a la pantalla de los demás, transferir archivos entre máquinas, enviar correo y conectarse en forma remota a otros computadores.
    - **Programas de aplicación**. Es normal que se entreguen con el S.O. programas como: generadores de gráficos, sistemas de Bases de datos, juegos... etc.

Repaso, conceptos importantes…

- Principales componentes del computador.
- Funcionamiento del computador. Ejecución de un programa en código máquina.
- Definición de sistema operativo.
- Niveles: hardware, sistema operativo, aplicación y usuario.
- Núcleo del sistema operativo.
- Tipos de sistemas operativos.
- Concepto de multiusuario y multitarea. Máquinas mono y multi-procesador.
- Componentes principales de la CPU.
- Modos de ejecución: kernel y usuario.
- Interrupciones: normales y por causas anómalas.
- Llamada al sistema. Instrucción TRAP. Cambio de modo de ejecución.
- Servicios básicos gestionados por el S.O.