Tema 1:

CARACTERIZACIÓN DE SISTEMAS INFORMÁTICOS.

FUNDAMENTOS DE LOS SISTE-MAS OPERATIVOS

1. EL SISTEMA INFORMÁTICO

Informática

Conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento automático de la información por medio de ordenadores.

Ordenador, computador o computadora

Máquina electrónica dotada de una memoria de gran capacidad y de métodos de tratamiento de la información, capaz de resolver problemas aritméticos y lógicos gracias a la utilización automática de programas registrados en ella.

Tratamiento de la información

entrada→ procesamiento → salida

planteamiento → análisis → programa (algoritmo)

Sistema informático

Es un conjunto formado por las 3 partes siguientes interrelacionadas:

1. Física: hardware
2. lógica:
   - software de base (firmware y sistema operativo):
     - **Firmware**: programa para propósitos específicos que controla los circuitos electrónicos de un dispositivo de cualquier tipo. Al estar integrado en la electrónica del dispositivo (en una memoria ROM) es en parte hardware, pero también es software. Es el intermediario (interfaz) entre las órdenes externas que recibe el dispositivo y su electrónica.
       - BIOS: firmware de la placa base del ordenador que se ejecuta al encenderlo. Una parte, el POST, reconoce y chequea los distintos componentes (procesador, memoria, teclado, discos duros, etc.). Otras partes se cargan en memoria para permitir al SO el acceso a los componentes. También contiene un programa de arranque (boot) que busca el programa de inicio del sistema operativo en los distintos dispositivos de almacenamiento (disco duro, CD-ROM,..)
     - Sistema operativo (SO): conjunto de programas que gestiona los recursos del sistema y **optimiza** su uso, actuando como intermediario entre el usuario y el hardware.
   - software de aplicación: programas de usuario, aplicaciones y suites
     - **Programa**: Conjunto de instrucciones que permite a un ordenador realizar funciones diversas para la resolución de una tarea.
     - una **aplicación** es un tipo de programa informático diseñado como herramienta para **permitir** a un usuario realizar uno o diversos tipos de trabajo.
     - Una suite (o paquete integrado) es una agrupación diversos programas y/o aplicaciones de distinta naturaleza
3. humano: desarrolladores (analistas y programadores), implementadores y usuarios

1.1. TIPOS DE SISTEMAS INFORMÁTICOS

- Según su uso:
  - De uso general
  - De uso específico (robots industriales, videoconsolas)
- Según sus prestaciones:
  - **Supercomputadoras**: gran capacidad de cálculo. Uso técnico-científico o simulaciones (Cray Jaguar posee en total 224.162 núcleos de procesamiento). Actualmente suelen ser cluster de ordenadores "blade". Se suele dar su rendimiento en gigaFLOPS.[1]
  - **Mainframes** (computadoras centrales): una computadora grande y potente usada principalmente por una compañía para el procesamiento de una gran cantidad de datos o dar soporte a grandes redes de comunicaciones. Su principal diferencia con las supercomputadoras es el manejo de grandes volúmenes de E/S
  - **Miniordenadores** con terminales "tontos" (dependen del miniordenador para la ejecución de procesos). En desuso.
  - **Estaciones de trabajo**: equipos monousuario de gran potencia y grandes prestaciones. Se diferencian de los microordenadores en que su hardware esta optimizado para ciertas tareas (diseño y CAD).
  - **Microordenadores**: equipos monousuario (PCs, sobremesa, portátiles).

---

[1] En informática, las **operaciones de coma flotante por segundo** son una medida del rendimiento de una computadora, especialmente en cálculos científicos que requieren un gran uso de operaciones de coma flotante. Es más conocido su acrónimo, **FLOPS**, por el inglés **floating point operations per second**. FLOPS, al ser un acrónimo, no debe nombrarse en singular como FLOP, ya que la S final alude a *second* (o segundo) y no al plural.

Las computadoras exhiben un amplio rango de rendimientos en coma flotante, por lo que a menudo se usan unidades mayores que el FLOPS. Los prefijos estándar del SI pueden ser usados para este propósito, dando como resultado megaFLOPS (MFLOPS, 10^6 FLOPS), gigaFLOPS (GFLOPS, 10^9 FLOPS), teraFLOPS (TFLOPS, 10^{12} FLOPS), petaFLOPS (PFLOPS, 10^{15} FLOPS), exaFLOPS (EFLOPS, 10^{18}FLOPS).

el rendimiento del equipo	
Nombre	**flops**
megaflops/megaofps	10^6
gigaflops/gigaofps	10^9
teraflops/teraofps	10^{12}
petaflops/petaofps	10^{15}
exaflops/exaofps	10^{18}
zettaflops/zettaofps	10^{21}
yottaflops/yottaofps	10^{24}

Nota: Actualmente, el uso general y específico está bastante difuso en algunos casos ya que por ejemplo una videoconsola permite conectarse a Internet entre otras cosas. Lo mismo pasa con las estaciones de trabajo y los microordenadores.

Investiga 1:

1. Nombra y explica las principales características y usos de 3 supercomputadoras.
2. Busca información sobre la Red española de supercomputación.
3. Nombra tres empresas/procesos que se realicen con mainframes.

2. EL SISTEMA OPERATIVO

El sistema operativo (SO) es un conjunto de programas que gestiona los recursos del sistema y optimiza su uso, actuando como intermediario entre el usuario y el hardware. Puede considerarse el sistema operativo desde dos puntos de vista:

- **SO como interfaz usuario-máquina**: Se encarga de que el usuario pueda abstraerse de las complejidades de funcionamiento del sistema, es una capa compleja entre el hardware y el usuario, que facilita herramientas adecuadas para realizar tareas informáticas, abstrayéndole de los complicados procesos necesarios para llevarlas a cabo.
- **SO como administrador de recursos**: debe repartir los recursos entre los distintos programas y y entre los usuarios de forma óptima, evitando tiempos muertos, controlando memoria, etc., pero como programa, el SO deberá cumplir los requisitos que el resto de programas que el usuario decida ejecutar, con la excepción de que tendrá más privilegios siendo el único con acceso a ciertos recursos del sistema

Los **objetivos** que debe cumplir el sistema operativo son los tres siguientes:

- **Eficiencia**: Permite que los recursos de un sistema informático se aprovechen al máximo, aumentando el rendimiento global del ordenador, por ejemplo pasando a otro programa cuando el que se está ejecutando se bloquea.
- **Comodidad**: Un sistema operativo facilita el uso del ordenador y debe hacerlo de la manera más cómoda para el usuario.
- **Capacidad de evolución**: Un sistema operativo debe construirse de modo que permita el desarrollo efectivo, la verificación y la introducción de nuevas funciones en el sistema sin interferir en los servicios que brinda, aprovechando las actualizaciones de nuevos tipos de hardware y ofreciendo nuevos servicios al usuario y la corrección de posibles fallos.

2.1. FUNCIONES DEL SISTEMA OPERATIVO

El SO es un programa más cuyas instrucciones ha de ejecutar el procesador. El SO dirige al sistema en el uso de los recursos y en el control de tiempo de ejecución asignado a cada programa.

Una parte del SO siempre está en memoria interna (esta parte se llama núcleo o kernel) e incluye las funciones utilizadas con mayor frecuencia. En un momento dado en la memoria puede haber otras partes del SO, además de programas y datos del usuario. La asignación de memoria también es controlada por el SO y el hardware de control de memoria del procesador.

El SO también decide cuándo puede utilizarse un dispositivo de E/S, y controla el acceso y utilización de archivos.

1. **Control de la ejecución de los programas (**gestión de procesos y gestión de interrupciones**).**

   Un proceso es una instancia de un programa que se está ejecutando, por lo que un mismo programa puede generar varios procesos. Estos procesos necesitan ciertos recursos, incluyendo tiempo de CPU, memoria, archivos y dispositivos de E/S.

El SO debe crear y terminar los procesos, así como suspenderlos y reanudarlos cuando sea necesario, asignándoles tiempo de procesador. Además de esto debe realizar el cambio de proceso de forma adecuada guardando toda la información del proceso que se suspende para poder seguir desde el mismo punto cuando se reanude.

Además, el SO debe proporcionar los mecanismos necesarios para la sincronización y comunicación entre procesos estableciendo prioridades cuando diferentes procesos soliciten el mismo recurso.

2. **Administración de periféricos** (gestión de la entrada-salida)

   El SO coordina y manipula los dispositivos conectados al ordenador. Cada dispositivo de E/S requiere un conjunto propio y peculiar de instrucciones y señales de control para su funcionamiento. El S.O. tiene en cuenta estos detalles de modo que el programador y el usuario puedan manejar el dispositivo de forma simple.

   - Gestión de Almacenamiento Secundario
     - Un **archivo** es una colección de información con nombre. Es la entidad básica de almacenamiento persistente.
     - El sistema de archivos suministra primitivas para manipular archivos y directorios: crear, borrar, leer, etc. También realiza la correspondencia entre archivos y su almacenamiento secundario abstrayendo al programador y al usuario de la naturaleza del dispositivo y del medio de almacenamiento y codificación de los datos.

3. **Gestión de permisos y de usuarios**

   Asigna los permisos de acceso a los distintos recursos (impresoras, carpetas, etc.).

4. **Administración de memoria**

   El S.O. asigna el espacio de direcciones a los procesos. Para ello debe gestionar la memoria de forma que divida toda la disponible entre el propio S.O. (recordemos que es un programa que se está ejecutando) y el resto de procesos de usuario. El S.O. debe mantener la pista de la de memoria utilizada actualmente y quién la usa. Debe además decidir cuanta memoria asignar a cada proceso, y cuando debe ser retirado de memoria un proceso.

5. **Funciones de soporte a las anteriores**

   - Intérprete de órdenes: Programa o proceso que maneja la interpretación de órdenes del usuario desde un terminal, o desde un archivo de órdenes, para acceder a los servicios del SO. Puede ser una parte estándar del SO (p. ej. el command.com de MS-DOS) o bien, un proceso no privilegiado que hace de interfaz con el usuario. Esto permite la sustitución de un intérprete por otro (p. ej. en Unix tenemos csh, bash, ksh, etc.).
   - Tratamiento de las interrupciones.
   - Detección y respuesta a errores: Cuando está funcionando un sistema pueden producirse errores internos y externos del hardware (errores de memoria, fallos en dispositivos, etc.) y/o errores de software (desbordamiento en una operación, acceso a zonas de memoria prohibidas o incapacidad del S.O. para satisfacer la solicitud de una aplicación). En cada caso el S.O. debe dar una respuesta que elimine la condición de error con el menor impacto posible. En las aplicaciones en ejecución, la respuesta puede ir desde terminar el programa que produjo el error hasta reintentar la operación o simplemente informar de tal error a la aplicación.

- Estadísticas: Un buen S.O. debe recoger estadísticas de utilización de los diversos recursos y supervisar los parámetros de rendimiento totales como puede ser el tiempo de respuesta ante la solicitud de una aplicación. Esta información puede resultar muy útil para mejoras futuras encaminadas a aumentar el rendimiento del sistema.
- Seguridad y protección del acceso al sistema: El S.O. controla el acceso al sistema informático como un todo y a sus recursos específicos. Puede brindar protección tanto a los recursos como a los datos ante usuarios no autorizados, y puede resolver conflictos en la propiedad de recursos.
- Creación de programas: Editores, depuradores (debugger), compiladores y linkadores ayudan en su tarea al programador. Normalmente estos servicios se hallan en programas de utilidad que no forman realmente parte del S.O. pero que son accesibles a través de él.

2.2. ESTRUCTURA DE UN SISTEMA OPERATIVO

Si estudiamos los sistemas operativos atendiendo a su estructura interna, veremos que existen dos tipos fundamentales: los sistemas de estructura monolítica y los sistemas de estructura jerárquica.

Estructura Monolítica

En los sistemas operativos de estructura monolítica el sistema operativo está formado por un único programa dividido en rutinas, en donde cualquier parte del sistema operativo tiene los mismos privilegios que cualquier otra. Cada una de las rutinas que forman el sistema operativo puede llamar a las demás cada vez que así lo requiera ya que todas son visibles entre ellas. El programa puede tener un tamaño considerable, y deberá ser recompilado por completo al añadir una nueva funcionalidad. Un error en una rutina puede propagarse a todo el núcleo. Todos sus componentes se encuentran integrados en un único programa que se ejecuta en un único espacio de direcciones. En este tipo de sistemas, todas las funciones que ofrece el sistema operativo se ejecutan en modo supervisor.

Estos sistemas operativos han surgido, normalmente, de sistemas operativos sencillos y pequeños a los que se les ha ido añadiendo un número mayor de funcionalidades. Esto les ha hecho evolucionar y crecer hasta convertirlos en programas grandes y complejos formados por muchas funciones situadas todas ellas en un mismo nivel.

El problema que plantean este tipo de sistemas radica en lo complicado que es modificar el sistema operativo para añadir nuevas funcionalidades y servicios. En efecto, añadir una nueva característica implica la modificación de un gran programa, compuesto por miles de líneas de código fuente y funciones.

Estructura Jerárquica

A medida que fueron creciendo las necesidades de los usuarios y se perfeccionaron los sistemas, se hizo necesaria una mayor organización del software del sistema operativo, donde una parte del sistema contenía subpartes y esto organizado en forma de niveles. Se dividió el sistema operativo en pequeñas partes, de tal forma que cada una de ellas estuviera perfectamente definida y con un claro interfaz con el resto de elementos. Se constituyó una **estructura jerárquica o de niveles o en capas** en la que se basan prácticamente la mayoría de los sistemas operativos actuales. Otra forma de ver este tipo de sistema es la denominada de anillos concéntricos o "*rings*"

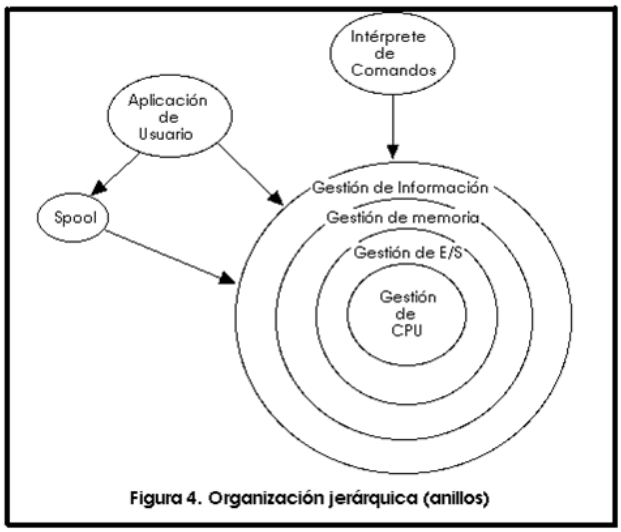


Figura 4. Organización jerárquica (anillos)

En el sistema de capas, cada capa es una máquina más abstracta para la capa superior. Cada una tiene una apertura, conocida como puerta o trampa (*trap*), por donde pueden entrar las llamadas de las capas inferiores. De esta forma, las zonas más internas del sistema operativo o núcleo del sistema estarán más protegidas de accesos indeseados desde las capas más externas. Las capas más internas serán, por tanto, más privilegiadas que las externas. Estos niveles son totalmente invisibles para el usuario y se puede decir que son los componentes internos del sistema operativo.

La división en capas no es la misma en todos los sistemas, así por ejemplo en **Windows NT** (2000, XP y posteriores**),** los niveles en que se divide el sistema operativo son los cuatro siguientes:

- Capa de abstracción de hardware (HAL, *Hardware Abstraction Layer*): Establece una correspondencia entre las órdenes y respuestas genéricas del hardware y aquellas que son propias de una plataforma específica, como un Intel 486 o un Pentium, un Power PC de *Motorola* o un procesador Alpha de *Digital Equipment Corporation*. La HAL hace que el bus del sistema de cada máquina, el controlador de DMA, el controlador de interrupciones, los relojes de sistema y el módulo de memoria parezcan los mismos para el núcleo.
- **Núcleo**: Consta de las componentes más usadas y fundamentales del sistema operativo. El núcleo administra la planificación de procesos, la gestión de excepciones (cepos) e interrupciones, la sincronización de multiprocesadores y la gestión de memoria virtual. En realidad se trataría de un microkernel (micronúcleo[2]).
- **Subsistemas**: Incluyen varios módulos con funciones específicas que hacen uso de los servicios básicos proporcionados por el núcleo y se ejecutan como procesos de usuario. Por ejemplo: proceso *log-on* y subsistema de seguridad.
- **Servicios del sistema**: Ofrece una interfaz al software en modo usuario.

En **UNIX**, el hardware básico está rodeado por el software del S.O. que se llama a menudo núcleo del sistema o, simplemente núcleo (*kernel*), para realzar su aislamiento de las apli-

2 La filosofía en la que se basa el micronúcleo es que sólo las funciones absolutamente esenciales del núcleo del sistema operativo deben permanecer en el núcleo. Las aplicaciones y los servicios menos esenciales se construyen sobre el micronúcleo como subsistemas externos que interactúan con éste (sistemas de archivos, de ventanas, servicios de seguridad, etc.) y que se ejecutan en algunos casos como procesos de usuario. El microkernel sólo necesitaría contener la funcionalidad básica para crear y comunicar procesos

caciones y de los usuarios. Sin embargo, UNIX viene equipado con una serie de servicios de usuario e interfaces que se consideran parte del sistema. Estos pueden agruparse en un *shell*[3], algún otro software de interfaz y los componentes del compilador de C. La capa exterior está formada por las aplicaciones de los usuarios y una interfaz de usuario con el compilador C.

2.3. UTILIZACIÓN DEL SISTEMA OPERATIVO

Debemos tener en cuenta también que externamente un S.O. se manifiesta a través de la interfaz de usuario, distinguiéndose a este respecto dos tipos:

- **Modo gráfico**: La interfaz está compuesta por un sistema de ventanas y menús desplegables que incluyen pequeños dibujos aclaratorios o iconos. Estos iconos pueden proporcionar un acceso directo a mandatos o aplicaciones. Esta interfaz es más compleja en su diseño pero mucho más fácil e intuitivo de utilizar para el usuario.
- **Modo comando**: Se basa en la introducción de comandos mediante el teclado en una línea identificada por un texto. Este identificador se conoce como prompt del sistema operativo. Este modo de uso de un SO sigue siendo imprescindible para la realización de muchas tareas de administración de sistemas, como puede ser la solución de problemas de arranque o la ejecución de ciertos procesos como la desfragmentación o el chequeo de discos duros. Se utiliza especialmente en administración Linux. Los sistemas operativos actuales suelen presentarse en modo gráfico, pero tienen acceso al modo comando.

2.4. CLASIFICACION DE LOS SISTEMAS OPERATIVOS

2.4.1 Por los servicios ofrecidos:

- Por el número de usuarios:
  - **Monousuario:** Los sistemas operativos monousuario son aquéllos que no soportan más de un usuario a la vez, sin importar el número de procesadores que tenga la computadora o el número de procesos o tareas que el usuario pueda ejecutar en un mismo instante de tiempo. Todos los recursos del sistema estarán disponibles para el único usuario posible.
  - **Multiusuario:** Los sistemas operativos multiusuario son capaces de dar servicio a más de un usuario a la vez, ya sea por medio de varias terminales conectadas a la computadora o por medio de sesiones remotas en una red de comunicaciones. No importa el número de procesadores en la máquina ni el número de procesos que cada usuario puede ejecutar simultáneamente.
- Por el número de tareas:
  - **Monotarea o monoprogramación:** Los sistemas monotarea son aquellos que sólo permiten una tarea a la vez por usuario. Puede darse el caso de un sistema multiusuario y monotarea, en el cual se admiten varios usuarios al mismo tiempo pero cada uno de ellos puede estar haciendo solo una tarea a la vez

3 El shell ofrece al usuario una interfaz con el sistema operativo separando al usuario de los detalles y presentándole el sistema operativo como un simple conjunto de servicios. El shell acepta las órdenes del usuario o las sentencias de control de trabajos, las interpreta, crea y controla los procesos según sea necesario.

- **Multitarea o multiprogramación:** Un sistema operativo multitarea es aquel que le permite al usuario estar ejecutando varios procesos al mismo tiempo. Por ejemplo, puede estar editando el código fuente de un programa durante su depuración mientras compila otro programa, a la vez que está recibiendo correo electrónico en un proceso en background (segundo plano). Estos procesos comparten tiempo de CPU hasta la finalización de cada uno de ellos. Es común encontrar en ellos interfaces gráficas orientadas al uso de menús y el ratón, lo cual permite un rápido intercambio entre las tareas para el usuario, mejorando su productividad.

- Por el número de procesadores que es capaz de usar:
  - **Monoproceso:** Un sistema operativo monoproceso o uniproceso es aquél que es capaz de manejar solamente un procesador de la computadora, de manera que si la computadora tuviese más de uno le sería inútil.
  - **Multiproceso:** Un sistema operativo multiproceso es capaz de manejar más de un procesador en el sistema, distribuyendo la carga de trabajo entre todos los procesadores que existan en el sistema
    - **Simétricos**: los procesos o partes de ellos (*threads*, hebras o hilos) son enviados indistintamente a cualquiera de los procesadores disponibles, teniendo, teóricamente, una mejor distribución y equilibrio en la carga de trabajo bajo este esquema.
    - **Asimétricos**: el sistema operativo selecciona a uno de los procesadores el cual jugará el papel de procesador maestro y servirá como pivote para distribuir la carga a los demás procesadores, que reciben el nombre de esclavos.

2.4.2 Por la forma de ofrecer los servicios:

- **Centralizados**: mainframes que se encargan de todo el procesamiento. Los usuarios se conectan a terminales que no procesan. Actualmente se siguen utilizando los sistemas centralizados (servidor Linux o Windows que ejecuta los servicios de terminal como Windows Terminal Services o Linux Terminal Services Project) pero los terminales ya sí procesan muchas tareas por sí mismos.

Distribuidos: un sistema distribuido es una colección de computadoras independientes que aparecen ante los usuarios del sistema como una única computadora. Los sistemas operativos distribuidos desempeñan las mismas funciones que un sistema operativo normal, pero con la diferencia de trabajar en un entorno distribuido. Su misión principal consiste en facilitar el acceso y la gestión de los recursos distribuidos en la red. En un sistema operativo distribuido los usuarios pueden acceder a recursos remotos de la misma manera en que lo hacen para los recursos locales. Permiten distribuir trabajos entre un conjunto de procesadores. Puede ser que este conjunto de procesadores esté en un equipo o en diferentes, lo cual es transparente para el usuario.

- **En red**. Varios ordenadores unidos a través de algún medio de comunicación, con el objetivo de compartir recursos e información. Cada ordenador tiene su propio sistema operativo.

- **De escritorio** o **S.O. cliente**: utilizados en equipos de sobremesa o portátiles.

2.4.3 Por su disponibilidad:

- **Propietarios**: Son propiedad intelectual de alguna empresa (Windows). Significa esto que se necesitan licencias de uso para ejecutarlos y no se tiene acceso a su código fuente, o si se tiene acceso, no se tiene derecho a modificarlo ni distribuirlo.
- **Libres**. Garantizan las cuatro libertades del software:
  1. Libertad de usar el programa con cualquier propósito.
  2. Libertad de estudiar cómo funciona el programa y modificarlo, adaptándolo a las necesidades que tuviera el usuario.
  3. Libertad de distribuir copias del programa, con lo que se puede ayudar a otros usuarios.
  4. Libertad de mejorar el programa y hacer públicas dichas mejoras a otros usuarios, de modo que toda la comunidad se beneficie de ello.

  El software libre es siempre de **código abierto** (libertades 2 y 4)

  El software libre no tiene por qué ser **software gratuito** ya que puede venderse una distribución con manuales, etc., en cuyo caso pasa a ser **software comercial**.

  El software gratuito (*freeware*) puede incluir el código fuente pero no tiene por qué ser libre a no ser que garantice el derecho de modificación y distribución de esas modificaciones.

  Software de **dominio público** es aquel que no requiere licencia pues todo el mundo puede usarlo pero no tiene por qué ser libre (puede no incluir el código fuente).

Investiga 2:

1. Identifica las características del SO de tu ordenador.
2. Nombra 2 sistemas operativos de cada tipo según los servicios ofrecidos.
3. Nombra un SO de cada tipo según la forma de ofrecer servicios.

3. TIPOS DE APLICACIONES Y TIPOS DE LICENCIA

En función del tipo de software las aplicaciones pueden ser:

a) **Gratuitas (*freeware*)** o **comerciales**.

b) **Libres** o **propietarias**. Libres se basan en la distribución del código fuente junto con el programa, así como en las cuatro premisas del software libre. Que una aplicación sea libre no implica que sea gratuita.

El software propietario es aquel en que los usuarios tienen limitadas las posibilidades de usarlo, modificarlo o redistribuirlo (con o sin modificaciones).

c) ***Opensource*** **(**código abierto) o **privativas** (código fuente no disponible o restringido).

Una **licencia** es un contrato mediante el cual un usuario recibe de una persona o empresa el derecho de uso de su software cumpliendo una serie de términos y condiciones establecidas dentro de sus cláusulas.

Algunos tipos de licencias del software propietario:

- **OEM**: La venta del software forma parte de un equipo nuevo
- **Retail**: El programa en este caso es enteramente del usuario que puede cederlo a terceros o venderlo.
- **Por volumen**: un número determinado de equipos pueden usar el mismo código de licencia. No se puede ceder a terceros.

El software libre está sujeto a su vez a una serie de licencias, cada una de ellas con sus respectivas normativas.

Investiga 3:

¿Qué implican los siguientes términos en cuanto a tipos de licencia?

GPL
BSD
Copyleft
Licencia de software libre
Licencia de software propietario
Freeware
Shareware
Opensource
EULA

Indica cuales de estos términos pueden referirse a un mismo producto y cuales son contradictorios.

4. LOS GESTORES DE ARRANQUE

Un **gestor de arranque** (en inglés *bootloader*) es un programa sencillo que no tiene la totalidad de las funcionalidades de un sistema operativo, y que está diseñado exclusivamente para preparar todo lo que necesita el sistema operativo para funcionar. Normalmente se

utilizan los cargadores de arranque multietapas, en los que varios programas pequeños se suman los unos a los otros, hasta que el último de ellos carga el sistema operativo.

En los ordenadores modernos, el proceso de arranque comienza cuando la unidad central de procesamiento ejecuta los programas contenidos en una memoria ROM en una dirección predefinida. El microprocesador se configura para ejecutar este programa, sin ayuda externa, al encender el ordenador. Este programa buscará en las unidades de almacenamiento del ordenador, otro programa que a su vez cargará el sistema operativo.

a. Windows XP, Windows Server 2003: NTLDR, boot.ini, NTDETECT.COM
b. Windows Vista, Windows 7, Windows Server 2008: Bootmgr, Winload.exe, ntowkrnl.exe
c. Linux: lilo o grub

Investiga 4:

Nombra dos gestores de arranque no asociados a ningún sistema operativo.

Trabajo voluntario:

Elabora una exposición sobre los gestores de arranque, incluyendo los arranques de Windows y Linux y algunos gestores comerciales.

5. GESTIÓN DEL PROCESADOR

5.1. Los procesos y la planificación

La mayoría de los procesadores dan soporte, al menos, para dos modos de ejecución. Estos son el modo usuario, el menos privilegiado, bajo el cual se ejecutan las aplicaciones del usuario, y el modo del sistema (o modo de control o del núcleo), el más privilegiado. Las instrucciones privilegiadas son aquellas que pueden potencialmente dañar a otros procesos, como por ejemplo desactivar interrupciones, la lectura o la modificación de los registros de control o aquellas instrucciones que se utilizan para gestionar la memoria; estas instrucciones sólo pueden ser llevadas a cabo por el núcleo del sistema.

La razón por la que se utilizan dos modos de ejecución radica en la necesidad de proteger al S.O. y a los datos importantes del mismo de injerencias de programas de usuario. En el modo núcleo, el software tiene el control completo del procesador y todas sus instrucciones, registros y memoria. Este nivel no es necesario, y por seguridad tampoco conveniente, para los programas de usuario.

5.2. Gestión de Procesos

Todos los sistemas operativos de multiprogramación están construidos en torno al concepto de proceso.

La misión principal del procesador es ejecutar las instrucciones que residen en memoria principal y que forman parte de un programa.

Desde el punto de vista del procesador, éste ejecuta instrucciones, dentro de un repertorio, según una secuencia dictada por los valores cambiantes de un registro: el contador de programa (PC) que no es más que una especie de puntero a las instrucciones. A lo largo del tiempo este valor puede apuntar a código de programas diferentes que son parte de diferentes aplicaciones. La ejecución de un programa individual se conoce como proceso o tarea. El comportamiento de un proceso individual viene determinado por el listado de la secuencia en que se ejecutan sus instrucciones. Dicho listado se llama traza del proceso.

En cuanto a los procesos, los requisitos principales que debe satisfacer un sistema operativo son:

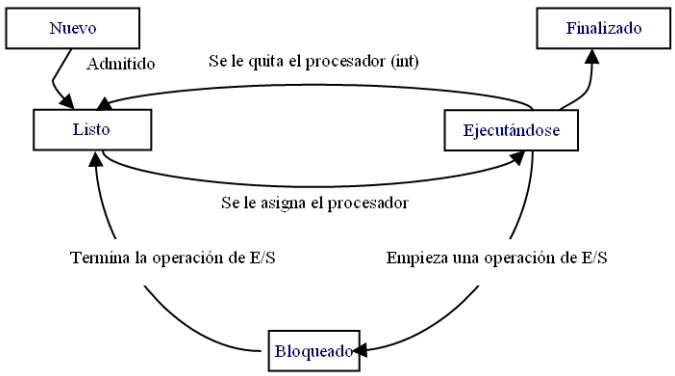
- El sistema operativo debe intercalar la ejecución de un conjunto de procesos para maximizar la utilización del procesador ofreciendo a la vez un tiempo de respuesta razonable.
- El sistema operativo debe asignar los recursos a los procesos en conformidad con una política específica (por ejemplo, ciertas funciones o aplicaciones son de prioridad más alta), evitando, al mismo tiempo el **interbloqueo**[4].
- El sistema operativo podría tener que dar soporte a la comunicación entre procesos y la creación de procesos por parte del usuario, labores que pueden ser de ayuda en la estructuración de las aplicaciones.

4 Un interbloqueo se produce básicamente si hay dos procesos que necesitan los mismos recursos para continuar y cada uno de ellos se ha apropiado de uno de los recursos. Cada proceso espera indefinidamente al recurso que le falta.

5.3. Estados de un proceso

Un proceso cargado en memoria se puede encontrar, en un momento determinado, en tres estados posibles:

- Ejecución (usando el procesador)
- Listo (proceso que está preparado para ejecutarse en cuanto se le dé la oportunidad).
- Bloqueado (proceso que no se puede ejecutar hasta que no se dé un determinado suceso, como puede ser la terminación de una E/S). Normalmente su ejecución no puede continuar porque ha de esperar datos de entrada que todavía no están disponibles.



Se puede considerar que un proceso está formado por los tres componentes siguientes:

- Código ejecutable del programa
- Los datos asociados necesarios para ejecutar el programa (variables, pila, buffers, etc.).
- El contexto de ejecución del proceso: atributos usados por el S.O. para el control del proceso (contenido del registro PC indicando la próxima instrucción a ejecutar y del resto de registros de la CPU, memoria utilizada, archivos abiertos, etc.)

Periódicamente, el S.O. decide parar la ejecución de un proceso y arrancar la de otro, por ejemplo, porque el primero ha consumido su tiempo de procesador. Cuando se suspende temporalmente la ejecución de un proceso, debe reiniciarse posteriormente en el mismo estado en que se encontraba cuando se paró. Esto implica que toda la información referente al proceso debe almacenarse explícitamente en alguna parte durante su suspensión. Por ejemplo, si el proceso tuviera varios ficheros abiertos, se debería guardar en algún sitio la posición exacta en donde se encontraba el proceso en cada fichero, de forma que una instrucción de lectura emitida después de que el proceso re-arrancase, leyera los datos apropiados.

El contexto de ejecución incluye toda la información que el sistema operativo necesita para administrar el proceso y que el procesador necesita para ejecutarlo correctamente. Esta información reside en el llamado bloque de control del proceso o descriptor de proceso. Diferentes sistemas organizarán esta información de diferente manera.

Veamos qué tipo de información podría usarse en el bloque de control de proceso en un sistema operativo, sin detallar la forma en que ésta se organizaría:

1. **Identificación del proceso**: Número identificador del proceso, del padre (si existe), del usuario, etc.
2. **Información del estado del procesador:** el contenido de los registros del procesador, que debe salvarse para poder restaurarse cuando el proceso reanude su ejecución.
3. **Información de Control del Proceso**: información adicional necesaria para que el S.O. controle y coordine los diferentes procesos activos: estado y prioridad de los procesos, dependencias padre-hijo, áreas de memoria asignadas, recursos en uso por cada proceso, etc.

Cuando se añade un proceso a los que ya está administrando el sistema operativo, éste debe crear las estructuras de datos que se utilizan para administrar un proceso y debe asignar el espacio de direcciones que va a utilizar (direcciones de memoria a las que puede acceder). Estas acciones constituyen la creación de un nuevo proceso.

5.4. Cambio de proceso

Un cambio de proceso puede producirse en cualquier momento en que el sistema operativo haya tomado el control a partir del proceso que esté ejecutándose. Los sucesos posibles que pueden dar el control al S.O. son una interrupción, un cepo o una llamada al supervisor. En primer lugar, vamos a tener en cuenta las interrupciones del sistema: una conocida simplemente como interrupción y otra como cepo.

Una interrupción es un mecanismo generado por algún tipo de suceso que es externo e independiente del proceso que está ejecutándose (como puede ser la culminación de una operación de E/S).

Un cepo tiene que ver con una condición de error o de excepción generada dentro del proceso que está ejecutándose (como puede ser un acceso ilegal a un archivo).

Una llamada al supervisor se produce cuando un proceso quiere realizar una operación para la que no tiene permisos, entonces cede el control a un módulo del SO.

En una interrupción ordinaria el control se transfiere primero al gestor de interrupciones quien lleva a cabo unas tareas básicas y, después, se salta a una rutina del sistema operativo que se ocupa del tipo de interrupción que se ha producido. Algunos ejemplos son los siguientes:

- **Interrupción de reloj**: El sistema operativo determina si el proceso que está en ejecución ha estado ejecutándose durante la fracción máxima de tiempo permitida. Si esto ocurre, el proceso debe pasar al estado listo y se debe pasar el control a otro proceso (ver Algoritmos de Planificación)
- **Interrupción de E/S**: Cuando el sistema operativo determina que se ha producido una operación de E/S y ésta constituye un suceso que han estado esperando uno o más procesos, modifica el estado de éstos de bloqueado a listo. Entonces deberá decidir si se reanuda la ejecución del proceso que está ejecutándose actualmente o si se le expulsa en favor de otro de mayor prioridad.
- **Fallo en memoria**: El procesador encuentra una referencia a una dirección de memoria virtual de una palabra que no se halla en memoria interna. El sistema operativo debe traer el bloque que contiene la referencia de la memoria secundaria a la principal. Después de hacer la solicitud para extraer el bloque de memoria secundaria el proceso que causó el fallo de memoria se pasa a estado bloqueado. Después de que el bloque en cuestión se cargue en memoria dicho proceso pasará al estado de listo.

En los cepos el sistema operativo determina si el error es fatal. En ese caso, el proceso que se estaba ejecutando se termina y se produce un cambio de proceso. Si por el contrario no se trata de un error fatal, la acción del sistema operativo dependerá de la naturaleza del mismo y del diseño del sistema operativo. Se puede intentar algún procedimiento de recuperación o, simplemente, reanudar el mismo proceso que estaba ejecutándose.

5.5. Planificación del procesador

El planificador de procesos es un elemento fundamental en los sistemas operativos multitarea. Se encarga de indicar qué proceso debe ejecutarse en cada momento y cuándo un proceso puede pasar de un estado a otro, según se dé un determinado suceso, como acabamos de ver.

El conjunto de criterios o reglas que sigue el planificador para decidir en cada momento qué proceso debe ejecutarse es lo que se denomina algoritmo de planificación.

El problema que plantea la planificación de procesos es que cada uno de ellos es único e impredecible. Algunos pasan mucho tiempo esperando operaciones de E/S y otros, aquellos con pocas o ninguna operación de este tipo, pasarían horas ocupando el procesador. El planificador no sabe, en principio, cuanto tiempo pasará hasta que un proceso termine o pase al estado de "bloqueado". Para controlar el tiempo que lleva en ejecución, activa una especie de cronómetro (un temporizador hardware, o reloj) que indica el tiempo de procesador consumido y produce interrupciones periódicas. Con cada interrupción de reloj, el sistema operativo recupera el control y decide si debe dejar que el proceso actual continúe ejecutándose, o si, por el contrario, éste ya ha utilizado el procesador durante un tiempo suficientemente largo y, por tanto, es el momento de suspender su ejecución y ceder el uso del procesador a otro proceso.

5.5.1 Tiempo Compartido

El concepto de tiempo compartido (*Time sharing*) hace referencia a compartir un recurso computacional entre muchos usuarios. Al permitir que un gran número de usuarios interactúe concurrentemente en una sola computadora, el coste del servicio de computación baja drásticamente, mientras que al mismo tiempo hace la experiencia computacional mucho más interactiva ya que cada usuario obtiene del ordenador un rendimiento casi igual al que obtendría si fuese el único en usarlo. De hecho, el usuario percibe la sensación de ser el único. Para llevar esto a cabo hace falta un algoritmo de planificación.

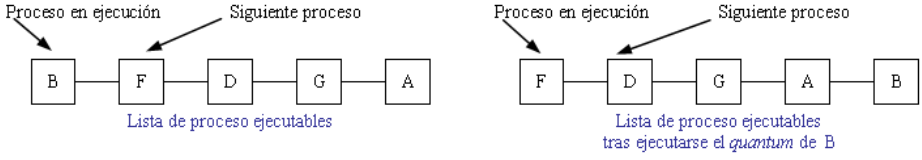
5.6. Algoritmos de Planificación

Multiprogramación clásica

Bajo esta técnica se cambia de un proceso a otro cuando el que está ejecutándose entra en una operación de E/S (o termina). Esto hace que procesos con mucho tiempo de cálculo/operaciones lógicas y pocas operaciones de E/S monopolicen la CPU.

Planificación por torneo, tiempo paralelo o turno rotatorio.

Consiste en crear una lista con los procesos pendientes de ejecutarse. A cada uno de ellos se le asigna un intervalo de tiempo fijo o periodo T, conocido como *quantum* o cuanto (usualmente del orden de décimas de segundo). Cuando un proceso finaliza su *quantum* el planificador pasa el control al siguiente programa y así sucesivamente. Cuando se acaba el *quantum* de un proceso, éste pasa al final de la lista.



Este sistema utiliza un circuito contador activado por el reloj del sistema (el cronómetro del que hablamos antes) que, cuando su salida llega a una determinada cuenta (ha transcurrido un tiempo de ejecución para un proceso igual al *quantum* que tenía asignado) provoca una interrupción de la CPU que es atendida por un módulo gestor de interrupciones y da el turno al siguiente trabajo a través del planificador. Si antes de que expire el *quantum* el proceso se bloquea (o ha terminado), el planificador da paso al siguiente proceso de la lista en el momento del bloqueo.

Planificación por prioridad.

A cada proceso se le asigna una prioridad de forma que siempre se concede el procesador al proceso ejecutable de más alta prioridad. Para evitar que los procesos de alta prioridad se ejecuten por tiempo indefinido, el planificador puede decrementar la prioridad del proceso que está ejecutándose cada cierto tiempo. Cuando su prioridad llegue a ser menor que la del proceso que le sigue, se realizará un cambio de proceso**.**

Lista de espera con intervalos múltiples.

Este algoritmo intenta beneficiar a los procesos que necesitan utilizar mucho el procesador. Para ello asigna intervalos de tiempo de ejecución diferentes según el proceso. El proceso empieza a ejecutarse con un *quantum* normal (igual para todos). Cuando ha terminado su *quantum* sin ser bloqueado, se incrementa éste para la siguiente vez. Cuando un proceso entra en estado "bloqueado", por ejemplo por la ejecución de una operación de E/S, vuelve a comenzar con el *quantum* inicial. Se debe establecer, no obstante, un *quantum* máximo para evitar que un proceso vaya aumentando el suyo hasta acaparar el procesador, paralizando el resto de los procesos.

Prioridad al más corto.

Especialmente indicado para sistemas por lotes en los que se conoce de antemano el tiempo de ejecución.

Ejemplo:

Tenemos 3 procesos (P1, P2 y P3) que se lanzan simultáneamente en tiempo=0.

- P1 necesita un total de 25 ms de microprocesador para finalizar, siendo de cálculo los primeros 15 ms y de E/S los otros 10 ms.
- P2 lee desde disco durante 3 ms, procesa durante 1 ms e imprime durante 6 ms. Así hasta 3 ciclos.
- P3 es sólo de cálculo y necesita 15 ms.

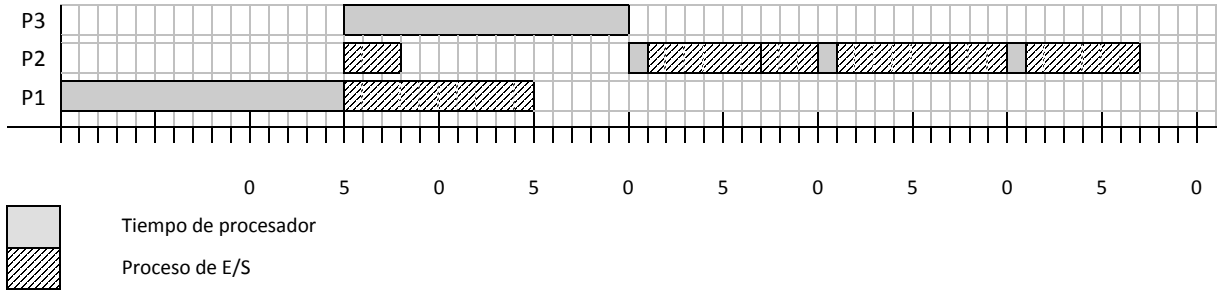
Calcular cuánto tiempo necesita en total para ejecutar los 3 procesos usando monoprogramación, multiprogramación clásica, multiprogramación en tiempo paralelo con quantum de10 ms y multiprogramación en tiempo paralelo con quantum de 5 ms.

Nota: en este ejemplo, y en los ejercicios, no consideraremos ni reflejaremos los tiempos en los que el sistema operativo toma el control del procesador, bien sea para realizar los cambios de proceso y/o para la gestión de interrupciones. Es decir, reflejaremos sólo el tiempo que el procesador invierte en ejecutar los procesos propiamente dichos.

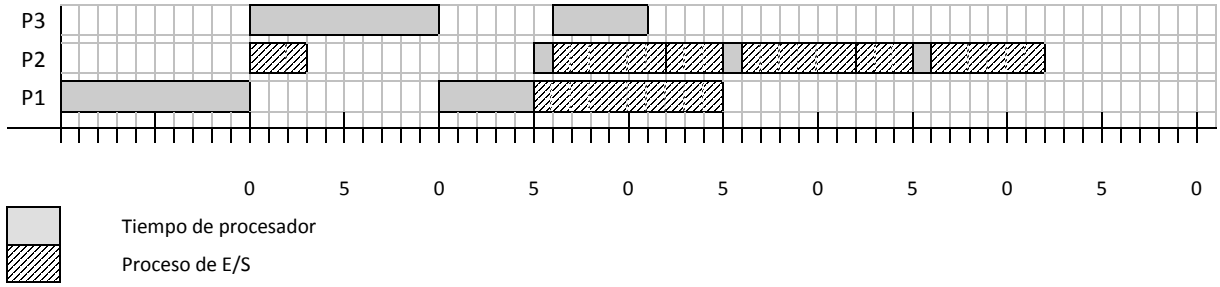
Solución:

a) monoprogramación: 25+[(3+1+6)*3]+15 = 70 ms

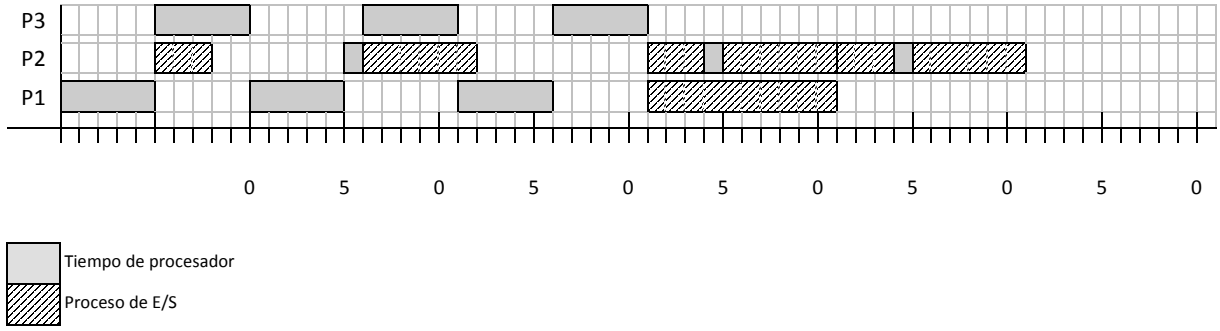
b) multiprogramación clásica: 57ms



c) multiprogramación tiempo paralelo con un quantum de 10 ms: 52 ms



d) multiprogramación tiempo paralelo con un quantum de 5 ms: 51 ms



6. GESTIÓN DE LA MEMORIA

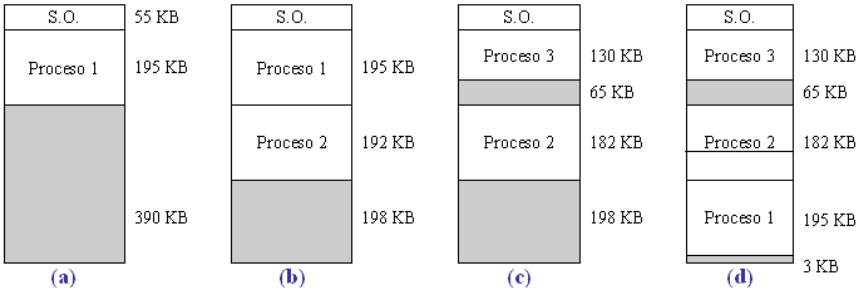
En un sistema monoprogramado, la memoria principal se divide en dos partes: una para el sistema operativo y otra para el programa de usuario que se ejecuta en ese instante.

En un sistema multiprogramación, la parte de "usuario" de la memoria debe subdividirse en más partes para dar cabida a varios procesos. La tarea de subdivisión la lleva a cabo dinámicamente el sistema operativo y se conoce como gestión de la memoria.

Se entiende que una gestión de la memoria efectiva es esencial en un sistema multiprogramado. Si sólo hay unos pocos procesos en memoria, la mayor parte del tiempo estarán esperando concluir sus operaciones de E/S y es posible que se produzcan tiempos muertos de CPU. Por ello, hace falta repartir eficientemente la memoria para albergar tantos procesos como sea posible.

Reubicación.

Es imposible que el programador conozca, de antemano, qué otros programas residirán en memoria en el momento en que se ejecute el suyo. El sistema operativo debe poder cargar y descargar los procesos en la memoria principal durante su tiempo de ejecución. Con esto se consigue maximizar el uso del procesador (más procesos listos para su ejecución) y una utilización más eficiente del espacio. Esto lo consigue mediante técnicas de intercambiabilidad memoria principal-disco (*swapping*). Para poder llevar a cabo estas funciones se requiere espacio para intercambio en almacenamiento secundario que puede ser una partición de disco o un fichero. El sistema operativo debe saber dónde situar un programa en concreto en la memoria y poder ubicarlo en zonas distintas si fuese necesario a resultas del mecanismo de *swapping*. Además en todo momento debe conocer la ubicación de la información de control del proceso.



(a) Se carga el Proceso 1, de 195 KB.

(b) Se carga otro, el Proceso 2 de 192 KB quedando libres 198 KB.

(c) El Proceso 1 entra en bloqueo y el sistema lo transfiere a disco. Entra el Proceso 3, de 130 KB, y ocupa parte del espacio que antes ocupaba el Proceso 1.

(d) El Proceso 1 debe volver a memoria pero el área que ocupaba anteriormente está ocupada por el Proceso 3. Si el sistema no fuese capaz de realizar la reubicación, el Proceso 1 no podría cargarse hasta que el Proceso 3 liberase dicha área de memoria. Con reubicación, el Proceso 1 se cargará en el área de memoria que aun queda libre, recalculando las referencias a memoria.

Además, el hardware del procesador y el software del sistema operativo deben ser capaces de traducir las referencias relativas a memoria encontradas en el código del programa a las direcciones físicas reales que reflejen la posición actual del programa en memoria principal.

Protección.

Cada proceso debe protegerse contra interferencias no deseadas de otros procesos, tanto accidentales como intencionadas. Así pues, el código de un proceso no puede hacer referencia a posiciones de memoria de otros procesos, con fines de lectura o escritura, sin permiso. La característica de reubicación de los procesos, sin duda, dificulta la protección de los mismos ya que se desconoce la ubicación de un programa en memoria principal y por tanto es imposible comprobar las direcciones absolutas durante la compilación para asegurar la protección. Por tanto, todas las referencias a memoria generadas por un proceso deben ser comprobadas en tiempo de ejecución para asegurarse que sólo hacen referencia a su zona de memoria.

No obstante, las exigencias de protección de memoria pueden ser satisfechas por el procesador (hardware) en vez de por el sistema operativo (software). Esto es debido a que el sistema operativo no puede anticiparse a todas las referencias a memoria que hará un programa. Incluso si esto fuera factible, sería prohibitivo en términos de tiempo consumido.

Compartición.

Cualquier mecanismo de protección debe tener la flexibilidad de permitir el acceso de varios procesos a una misma zona de memoria principal compartida. Por ejemplo, si una serie de procesos están ejecutando el mismo programa sería beneficioso permitir que cada proceso accediese a la misma copia del programa en lugar de tener una por cada uno de ellos.

6.1. Técnicas de Gestión de Memoria

6.1.1 Particiones Estáticas

Es la forma más sencilla de organizar la memoria para poder tener varios procesos simultáneamente residentes en memoria. Consiste en dividir la memoria en n particiones, quizás de tamaño diferente, cada una de las cuales contendrá un programa. Las direcciones base[5] son las direcciones de comienzo de cada partición. El tamaño de cada partición es determinado por el operador o por el sistema operativo por lo que el tamaño de las particiones ya se sabe antes de cargar los programas en memoria. Normalmente, el número de particiones y su tamaño no puede cambiarse una vez establecido en una sesión de trabajo por lo que debería reiniciar el sistema para cambiarlas.

Algoritmo de ubicación.

El sistema operativo mantiene una tabla en la que cada fila corresponde a una partición, conteniendo la posición base de la misma, su tamaño (no todas las particiones tienen por qué ser iguales) y el estado de la partición (ocupada o libre). El planificador de procesos, una vez que una partición queda libre, hace que se introduzca el programa de máxima prioridad que quepa en ella.

Este método está hoy ampliamente superado y presenta básicamente dos problemas:

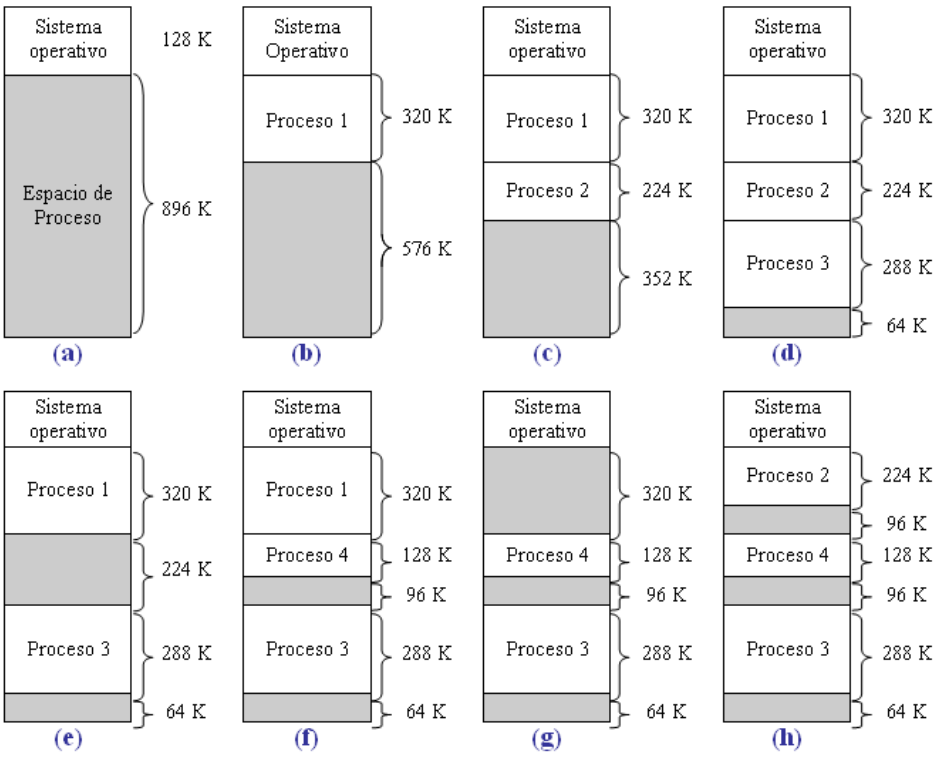
- Un programa puede no caber en una partición, lo que obliga al programador a estructurar su aplicación en módulos que sí quepan.

[5] Dirección donde se carga la primera instrucción del proceso y, a partir de la cual, se calculan el resto de las direcciones que utiliza el proceso

- Efectúa un uso ineficiente de la memoria ya que cualquier programa, por pequeño que sea, ocupa una partición entera lo que provoca una fragmentación "interna": se malgasta el espacio interno de una partición.

6.1.2 Particiones Dinámicas

Las particiones son variables en número y longitud. Conforme se carga un proceso se le asigna la memoria que necesita y no más. En la siguiente figura se muestra un ejemplo que usa 1MB de memoria principal. Al principio, la memoria principal está vacía, exceptuando el sistema operativo.



Efectos de la partición dinámica

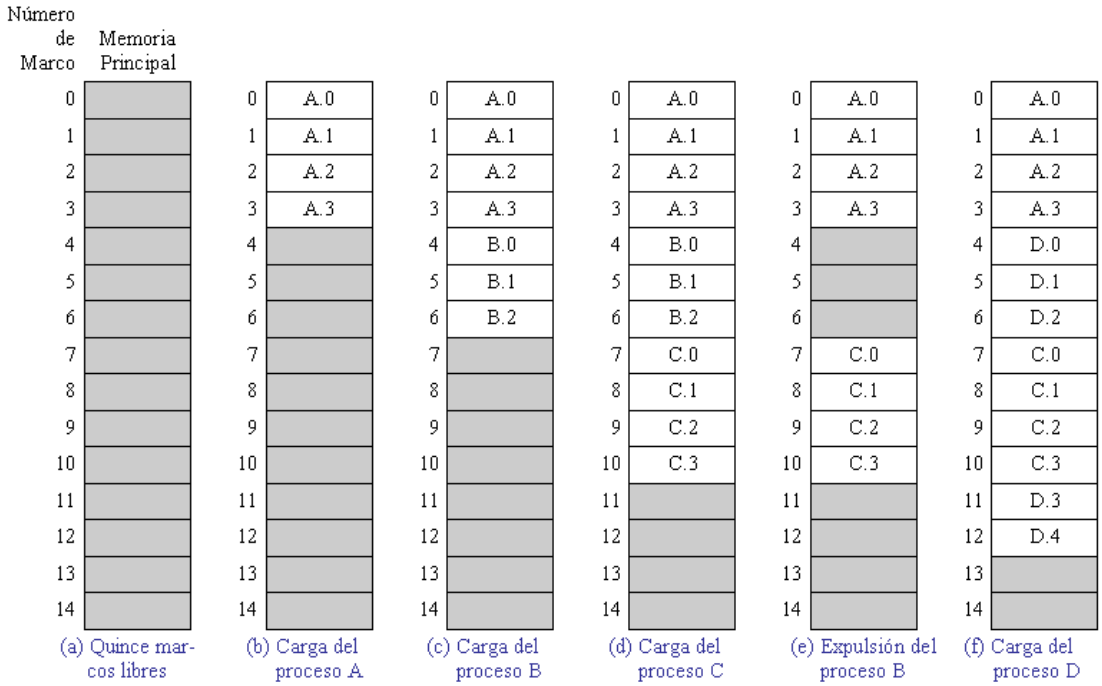
Como se ve, este método comienza bien, pero finalmente, desemboca en una situación en la que hay un gran número de huecos pequeños en memoria, problema que puede resolverse mediante la compactación de los huecos libres. La compactación se puede efectuar cambiando de lugar los programas en ejecución aprovechando la posibilidad de reubicarles que debe ofrecer el sistema operativo.

6.1.3 Paginación

Con este procedimiento, la memoria principal se estructura en trozos de tamaño fijo llamados marcos o marcos de página o simplemente bloques. Cada proceso está dividido también en pequeños trozos de tamaño fijo y del mismo tamaño que los marcos de memoria llamados páginas. De esta forma, las páginas pueden asignarse a marcos de página libres.

El fundamento de la paginación reside en que no es necesario que un programa se almacene en posiciones consecutivas de memoria. Las páginas se almacenan en marcos libres, independientemente de que estén o no contiguos. Eso sí, cada bloque contendrá, obviamente, instrucciones consecutivas.

Una instrucción de programa se localiza dando el marco de memoria y la dirección relativa dentro del bloque.

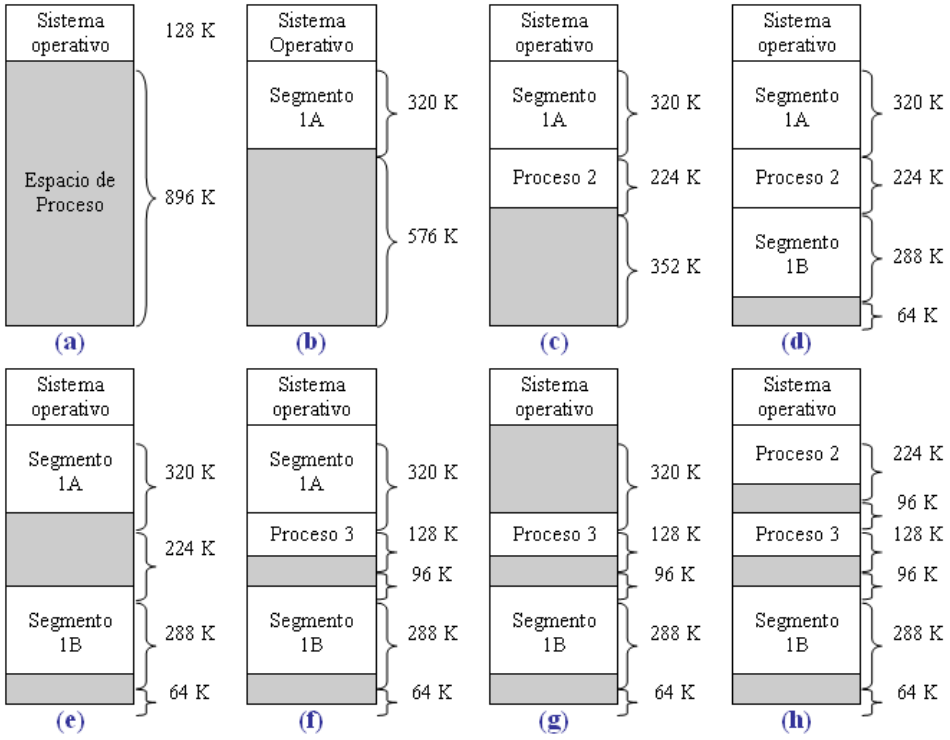


Asignación de páginas de procesos a marcos libres

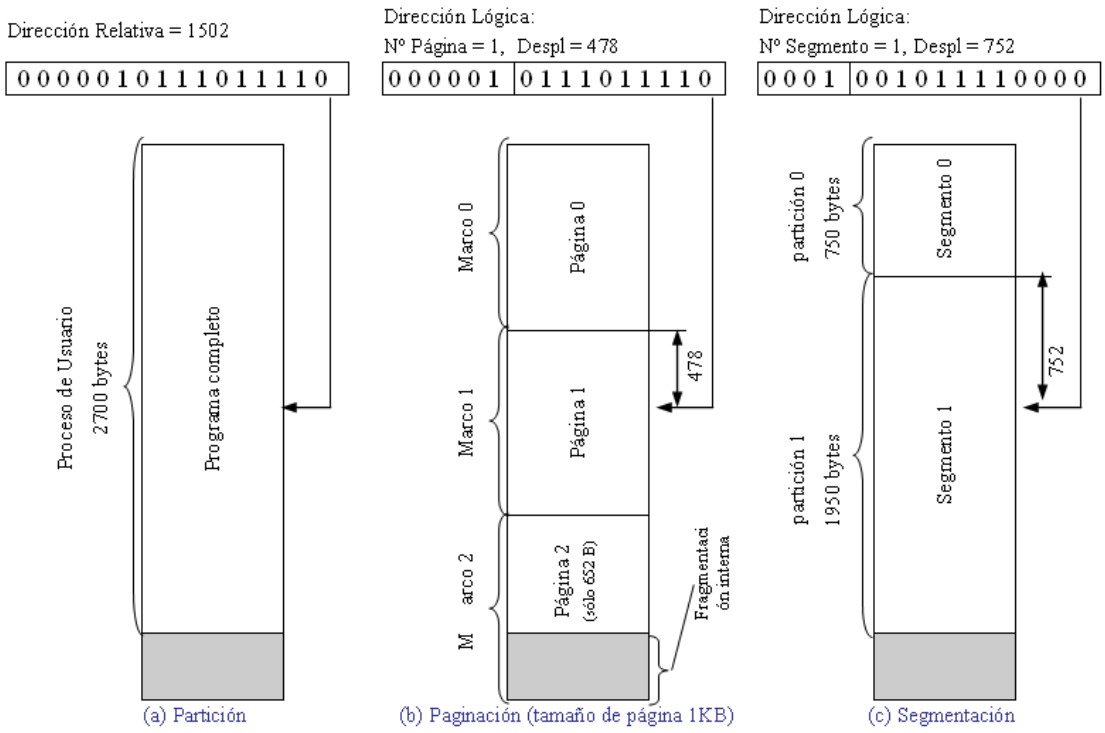
6.1.4 Segmentación

El programa se considera divido en segmentos. Hay que tener en cuenta que nos referimos a los segmentos, en este caso, como a grupos lógicos de información (código, pila, datos, etc.) definibles por el programador o asignados por el compilador, de forma que se puede decir que un programa es una colección de segmentos. Como en paginación, una dirección lógica segmentada consta de dos partes, en este caso un número de segmento y un desplazamiento.

La gestión la realiza el sistema operativo como en el caso de las particiones dinámicas, sólo que cada partición no corresponde a un programa sino a un segmento del mismo y las particiones que ocupa un programa no tienen por qué estar contiguas.



Reubicación de la segmentación



Direcciones lógicas

6.2. Memoria Virtual

La memoria virtual permite a los usuarios hacer programas de una capacidad superior a la que físicamente tiene la computadora. Los dos enfoques básicos de memoria virtual son la paginación y la segmentación, aunque también se pueden combinar ambos enfoques en un

único esquema de gestión de memoria. La memoria virtual exige un soporte tanto de hardware como de software. El soporte hardware lo proporciona el procesador e incluye la traducción dinámica de direcciones virtuales a direcciones físicas y la generación de interrupciones cuando una página o segmento referenciado no están en memoria principal. Estas interrupciones activan el software de gestión de memoria del sistema operativo.

Primero vamos a centrarnos en los aspectos del hardware de la memoria virtual, considerando el uso de la paginación, la segmentación o una combinación de ambas. Tanto una como otra técnica presentan dos características:

1.- Todas las referencias a memoria dentro de un proceso se hacen por medio de direcciones lógicas que se traducen dinámicamente en direcciones físicas durante la ejecución. Esto facilita la reubicación de un proceso si fuese necesario.

2.- Un proceso puede dividirse en partes, páginas o segmentos, y no es necesario que estas partes se encuentren consecutivas en la memoria principal durante la ejecución. Esto es posible por la combinación de la traducción dinámica de direcciones en tiempo de ejecución y el uso de una tabla de páginas o de segmentos.

La consecuencia inmediata de estas ventajas es que no es necesario, por tanto, que todas las páginas o todos los segmentos de un proceso estén en memoria durante la ejecución. Basta con que estén los segmentos o páginas donde se encuentren las siguientes instrucciones a ejecutar o la siguiente zona de datos a la que se tiene que acceder.

Esto puede suponer mejoras en la utilidad del sistema puesto que:

1.- Se pueden conservar más procesos en memoria puesto que hay más espacio disponible. Esto implica un mayor aprovechamiento del procesador al haber menos probabilidad de que todos los procesos en ejecución estén en un momento determinado en estado “bloqueado” (esperando una E/S).

2.- Es posible ejecutar procesos muy grandes e, incluso, un único proceso puede superar la capacidad de la memoria ya que el sistema operativo sólo cargará en memoria interna los fragmentos que necesite de él en un momento determinado, permaneciendo el resto en memoria secundaria hasta ser requeridos.

Como los procesos se ejecutan sólo en memoria principal, a esta memoria se le denomina memoria real. Pero un programador o un usuario percibe en potencia una memoria mucho mayor que está situada en el disco, es decir, podemos hablar de una memoria virtual.

El fin que se persigue es que en un instante dado, en memoria sólo se tengan unos pocos fragmentos de un proceso y, por tanto, poder mantener más procesos en memoria. Es el sistema operativo el encargado de mantener este esquema. Pero esto puede hacer necesario el tener que evacuar un fragmento para hacerle sitio a otro que se necesite para la ejecución. Además, si se expulsa un fragmento que seguidamente va a ser utilizado, obligatoriamente deberá el sistema volver a cargarlo casi de forma inmediata. En definitiva, el sistema implica un trasvase de fragmentos desde memoria secundaria hasta memoria principal. Esto podría suponer un hándicap que hiciera ineficaz esta gestión de la memoria, puesto que demasiadas operaciones de este tipo pueden conducir a lo que se conoce como *trashing* (hiperpaginación), es decir, que el procesador consuma más tiempo en el intercambio de fragmentos que ejecutando instrucciones.

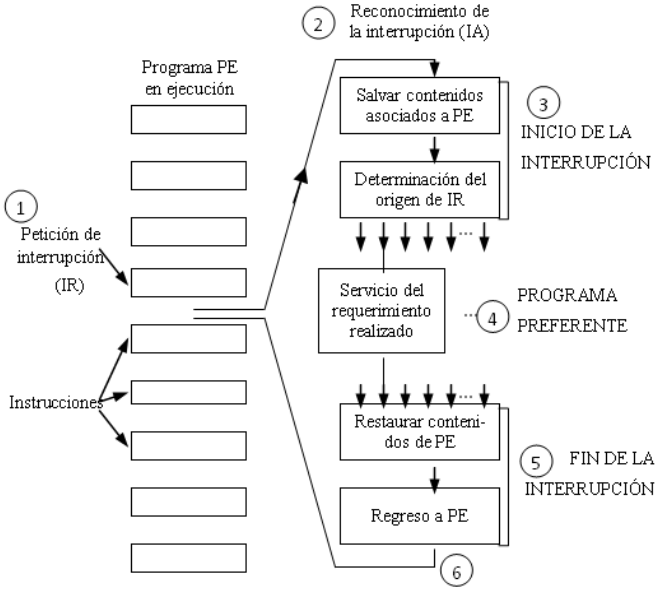
7. GESTIÓN DE LAS INTERRUPCIONES.

Las interrupciones son un método del que disponen los elementos hardware (E/S, memoria, reloj) e incluso los procesos para hacer notar a la CPU la aparición de alguna circunstancia que requiera su intervención.

Se utilizan las interrupciones generalmente por dos motivos: permitir una comunicación sin bloqueo con los periféricos externos y conmutar las tareas dentro de un planificador de procesos (como hemos visto en el apartado correspondiente).

Las interrupciones de E/S son generadas por un controlador de E/S para indicar que una operación ha terminado normalmente o para indicar diversas condiciones de error.

El acontecimiento de una interrupción desencadena una serie de sucesos, tanto en hardware como en el software del sistema.



Esquema que muestra la gestión de una interrupción de un programa PE para atender a otro

1.- Petición o demanda de interrupción (*interruption request*): La petición se realiza por medio de una señal eléctrica (señal de estado) enviada a la CPU.

2.- Por lo general, la CPU no atiende inmediatamente la petición de la interrupción, sino que antes acaba de ejecutar la instrucción que se halla en curso. Por medio de instrucciones adecuadas, las CPU actuales pueden inhibir las peticiones de interrupción cuando se ejecuta un módulo o programa de alto privilegio, como el núcleo del sistema operativo, por ejemplo, y posteriormente habilitar o atender las peticiones que mientras tanto se hubiesen producido. También hay técnicas de arbitraje de peticiones de interrupción, de forma que cuando hay varias peticiones pendientes se prevé cuál atender primero.

3.- Se atiende la interrupción (*interruption acknowledge*) poniéndose en marcha una rutina o programa de servicio de inicio de interrupción que analiza la causa de la interrupción, salva el contenido de los acumuladores/registros de la CPU (contador de programa, PSW, etc.) y pasa el control al programa preferente. La salvaguarda de estos elementos se hace para que no se pierdan con la ejecución del programa preferente.

4.- Se ejecuta la rutina o programa preferente, que atiende el requerimiento de la interrupción.

5.- Se ejecuta la rutina de fin de servicio de interrupción que restaura el contenido de los registros de la CPU.

6.- La rutina de fin de servicio de interrupción pasa el control de ejecución al programa interrumpido, continuándose éste.

Hay que tener en cuenta que un programa que interrumpe a otro puede ser, a su vez, interrumpido y así sucesivamente.

8. GESTIÓN DE LA E/S

La gestión de la E/S es uno de los aspectos más confusos en el diseño de los SO. Las características de los dispositivos de E/S suelen diferir mucho de las de la CPU, por ejemplo en la velocidad, ancho de palabra, etc. Aparte de esto, la gran variedad de dispositivos impide construir una solución general para la gestión de los mismos.

Normalmente las unidades de E/S constan de dos elementos: mecánico, que es el dispositivo propiamente dicho y electrónico al que se le suele denominar controlador. El sistema operativo casi siempre trata con el controlador.

Todos los controladores disponen de unos cuantos registros para comunicar con el procesador. Estos registros pueden formar parte del espacio normal de direcciones de memoria (tienen un trozo de memoria reservada para esta función). En otros casos se utiliza un espacio de direcciones especial para E/S, con una parte del mismo asignada a cada controlador. Para realizar operaciones de E/S, el sistema operativo escribe comandos en los registros del controlador, el controlador acepta el comando y mientras el procesador puede dejarlo que haga su trabajo solo y ocuparse entretanto de otras cosas.

8.1. Técnicas de E/S

E/S Programada

Con esta técnica, ya obsoleta, el módulo de E/S no avisa al microprocesador de nada en particular, es decir, no lo interrumpe. Es el microprocesador el responsable de comprobar periódicamente el estado del módulo de E/S hasta saber que se ha completado una operación y cual es el resultado (comprobar por ejemplo si ha habido un error).

Es también el microprocesador el responsable de extraer los datos de la memoria interna cuando se va a hacer una salida y de cargarlos en ella en el caso de una entrada. El software está diseñado de tal forma que se le otorga al microprocesador el control directo sobre la operación de E/S incluyendo la comprobación del estado del dispositivo, el envío de órdenes de lectura/escritura y la transferencia de los datos. Se trata de un proceso que consume tiempo y mantiene ocupado al procesador de forma innecesaria.

E/S Dirigida por interrupciones

Con esta técnica el microprocesador envía una orden de E/S al módulo correspondiente y mientras se dedica a hacer otra cosa, sin tener que comprobar el estado del módulo de E/S cada cierto tiempo. El módulo de E/S interrumpirá al microprocesador cuando necesite intercambiar información con él. El microprocesador obtiene los resultados de la operación y el estado del dispositivo leyendo uno o más bytes de información de los registros del controlador mencionados anteriormente.

Se sigue consumiendo tiempo de microprocesador debido a que cada palabra de datos que va de la memoria al módulo de E/S o viceversa debe pasar por él.

DMA

Muchos controladores tienen acceso directo a memoria (DMA, *Direct Memory Access*), útil sobre todo cuando se tienen que mover grandes cantidades de información.

La técnica funciona de la siguiente forma: cuando el microprocesador desea leer o escribir un bloque de datos emite una orden para el módulo DMA y le envía la información siguiente:

- Si lo que solicita es una lectura o escritura
- La dirección del dispositivo de E/S involucrado
- La dirección inicial de memoria a partir de la cual se va a leer o escribir
- El número de palabras a leer o escribir

Seguidamente el microprocesador continúa con otro trabajo y mientras el módulo DMA se ocupa de la E/S, transmitiendo la información desde o hacia la memoria sin pasar por el microprocesador. Cuando se completa la transferencia, el módulo DMA envía una señal de interrupción al microprocesador. De esta forma el microprocesador se ve involucrado en la E/S sólo al principio y al final de la operación.

Si no existiera DMA, para leer datos de un disco por ejemplo, el controlador tendría que leer el bloque de la unidad de disco y llevarlo al buffer interno del controlador. Después generaría una interrupción. Cuando el sistema operativo ejecutara la rutina de interrupción, podría leer el bloque de disco del buffer del controlador palabra a palabra y escribirla en la memoria. Cada palabra leída del controlador necesitaría tiempo de procesador. El DMA se inventó para liberar al procesador de esta actividad de bajo nivel.

8.2. Almacenamiento intermedio de la E/S: *Buffering*

Para evitar las consecuencias que se derivan de la diferencia en la velocidad de funcionamiento de los dispositivos y la CPU se suele utilizar un buffer. Un **buffer** es un almacén de información. El buffer del controlador se utiliza para guardar temporalmente los datos implicados en una operación de E/S. Por ejemplo, si se quiere escribir en una impresora, se carga la información a escribir desde memoria principal al buffer. Posteriormente, el controlador mandará dicha información desde el buffer a la impresora. Si se trata de una unidad de entrada, la transferencia de información se puede hacer por adelantado de forma que la CPU tenga los datos en el buffer cuando los necesite.

El almacenamiento intermedio es una técnica que soluciona los problemas de "horas punta" en la demanda de E/S. Sin embargo, no existe un tamaño de los buffers que asegure a un dispositivo de E/S ir al mismo ritmo que un proceso cuando la demanda media del proceso es mayor que la que el dispositivo puede admitir. Incluso si se dispone de varios buffers, al final todos se llenarán y el proceso tendrá que quedarse esperando tras operar con una determinada cantidad de datos. Sin embargo, en un entorno de multiprogramación, con la variedad de actividades de E/S y de procesamiento que hay que realizar, el almacenamiento intermedio es una herramienta que puede incrementar la eficiencia del sistema operativo y el rendimiento de los procesos individuales.