

CONTENIDO

	<i>Página</i>
INTRODUCCION	1
OBJETIVOS	4
UNIDAD 1. TEORIA DE LOS SISTEMAS OPERATIVOS	5
Introducción	5
Objetivos	6
Capítulo 1. Principios de los sistemas operativos	7
1. Concepto de sistema operativo	7
2. Estructura de un sistema computacional	9
3. Historia de los sistemas operativos	21
4. Conceptos de sistemas operativos	25
Capítulo 2. Características de los sistemas operativos	30
1. Funciones de los sistemas operativos	30
2. Tipos de sistemas operativos	32
3. Estructura de los sistemas operativos	46
4. Núcleos de sistemas operativos	49
Capítulo 3. Interbloqueo	53
1. Recursos	53
2. Análisis	56
3. Condiciones para producir un interbloqueo	60
4. Métodos para manejar el interbloqueo	61
5. Prevención del interbloqueo	62
6. Evasión del interbloqueo	66
7. Mecanismos para evitar el interbloqueo	67
8. Nivel de implantación de estrategias	74
9. Detección y recuperación	76
Capítulo 4. Arquitectura cliente servidor	82
1. Antecedentes	82
2. Cliente /servidor	85
3. Componentes esenciales de la infraestructura cliente/servidor	87
4. Características funcionales	88
5. Características físicas	90
6. Características lógicas	91
7. Middleware robusto y escalable en soluciones Cliente/Servidor	91
8. Análisis de las diferentes variantes de la arquitectura C/S	95
9. Arquitecturas Cliente/Servidor independientes de plataforma	97
10. Condiciones para la implantación del modelo Cliente/Servidor	100
11. Costos y beneficios de Cliente/Servidor	101
12. Fases de implantación	103
Bibliografía	107

UNIDAD 2. ADMINISTRACION DE RECURSOS	109
Introducción	109
Objetivos	110
Capítulo 1. Administración de los procesos	111
1. Introducción a los procesos	111
2. Hebras de control (Threads)	119
3. Comunicación entre procesos	121
4. Sincronización de procesos	125
5. Planificación de procesos-Scheduling-Itineración de procesos	130
Capítulo 2. Administración de la memoria	142
1. Estructura general	142
2. Manejo de memoria en sistemas monousuario sin intercambio	142
3. Multiprogramación en memoria real	143
4. Multiprogramación en memoria virtual	147
5. Algoritmos de reemplazo de páginas	161
Capítulo 3. Administración de dispositivos	169
1. Entrada/Salida	169
2. Dispositivos de Entrada/Salida	170
3. Controladores de Dispositivos (Terminales y Discos Duros)	171
4. Principios en el Software de Entrada - Salida	172
5. Relojes	173
6. Puertas y buses	173
7. Protocolos de interacción	175
Capítulo 4. Administración de archivos	178
1. El sistema de archivos	178
2. Métodos de acceso	180
3. Directorios	182
4. Protección	184
5. Estructura de archivos	186
6. Métodos de asignación	188
7. Tipos de sistemas de archivo	193
8. Mecanismos de protección de los ficheros	195
9. Administración del espacio libre	198
Capítulo 5. Protección y seguridad	200
1. Protección	200
2. Objetivos de la protección	200
3. Dominios de la protección	201
4. Matriz de acceso	203
5. Seguridad	204
6. Autenticación	205
7. Amenazas	206
8. Encriptación	208
9. Clasificación de seguridad	211

UNIDAD 3. PRINCIPALES SISTEMAS OPERATIVOS	214
Introducción	214
Objetivos	216
Capítulo 1. Sistemas operativos Familia Windows	217
1. Windows 95	218
2. Windows 98	221
3. Windows 2000	227
4. Windows Server 2003	231
5. Windows XP	236
6. Windows NT	243
7. Instalación Windows XP/2003	255
8. Resumen de los principales sistemas operativos Windows	265
Capítulo 2. Sistema operativo UNIX/LINUX	267
1. Historia	267
2. Arquitectura de Unix y Linux	269
3. Versiones y características Linux/Unix	277
4. Instalación del sistema	281
5. La interfaz de usuario	292
6. Estructura de archivos	302
7. Variables de entorno	307
8. Administración del sistema	308
9. Montando sistemas de ficheros	317
10. Instalación de impresoras y gestión de la impresión	320
11. Resumen de órdenes básicas	324
Capítulo 3. Otros sistemas operativos	328
1. Novell Netware	328
2. Sistema operativo OS/2	336
3. El sistema operativo VMS (Virtual Memory System)	341
Capítulo 4. Arquitectura de las comunicaciones	346
1. El modelo arquitectónico de capas de red	346
2. El modelo de referencia OSI	347
3. Otras arquitecturas y redes	349
Bibliografía	357

INTRODUCCIÓN

Este módulo está diseñado para aquellas personas interesadas en abordar temáticas relacionadas con el campo de los sistemas operativos: teoría y aplicación. Trata de acercarlas a los aspectos más importantes que encierran los sistemas operativos, ubicando características básicas tanto a nivel de PC como a nivel de infraestructura de red.

Por ello y para ello, se presenta el módulo “Sistemas Operativos”, como una alternativa de solución para esa búsqueda de enfoques multidisciplinarios, y por lo tanto, aplicables a cualquier sistema que se desee analizar o implementar.

Así, teniendo en cuenta que en absolutamente cualquier área de la ingeniería de sistemas en la que nos desempeñemos o queramos desempeñarnos vamos a trabajar con sistemas operativos: implementandolos, administrándolos, diseñandolos, desarrollandolos, utilizandolos, enseñandolos y resolviendo problemas que ahora serán más fáciles de solucionar, pues uno de los objetivos primordiales es el de estudiar a fondo su estructura para eliminar cualquier complejidad que tengamos con respecto al tema.

El sistema operativo es una parte fundamental de cualquier sistema computacional, lo que nos lleva a confirmar, aún más, la importancia de su conocimiento y manejo, y más en nuestra formación como ingenieros de sistemas.

Es bueno aclarar que no es un módulo orientado hacia la guía e instalación de algunos sistemas operativos. Considerando que estamos trabajando en el ciclo profesional, no se torna relevante, por varias razones:

- Ahora con la gama de asistentes gráficos disponibles para la instalación, este proceso se torna en tarea sencilla, hasta para un usuario novato.
- Se ha pasado por una serie de cursos en los que ya han trabajado, la instalación de sistemas operativos, por lo menos, para computadores personales.

Al contrario, es un módulo que brinda toda la documentación e información relativa a las características, estructura, diseño, componentes que ayudan a conocer de forma integral un sistema operativo, para que podamos administrarlo de la mejor manera y sacar el máximo provecho a todas sus capacidades e incluso podamos enfrentarnos al reto de diseñar o adecuar nuestro propio sistema operativo.

Con el fin de afianzar el aprendizaje de los contenidos, así como el de las habilidades, a lo largo de los capítulos se incluyen ejercicios y/o ejemplos que sirven como activación cognitiva, para ubicar a los interesados en el contexto a desarrollar, también en algunos casos para reforzar o reafirmar una temática y al final de cada capítulo se encuentran ejercicios que servirán para la transferencia de los contenidos desarrollados a las diferentes prácticas de laboratorios ó a situaciones cotidianas o laborales y a sus intereses tanto profesionales como personales.

Los ejercicios propuestos vienen diseñados para que se resuelvan de manera individual, como actividad complementaria o para resolverlo en grupos de trabajo y así profundizar en los contenidos relacionados y para desarrollar habilidades como comunicación oral, comunicación escrita y trabajo colaborativo.

Este módulo es el resultado de un trabajo extenso de consulta, investigación bibliográfica y sistematización de experiencias, el cual sirvió para la consolidación de la información, contenidos temáticos y ejercicios con el fin de brindar, además, una herramienta de consulta apropiada al curso académico, ala metodología de trabajo y a las necesidades que pretende cubrir cada persona.

Por ello en cada unidad didáctica se encuentra una sección bibliográfica recomendada, incluyendo direcciones de Internet con las que se puede ir más allá en el logro de los objetivos propuestos.

El desarrollo temático de los capítulos contempla, intrínsecamente, la articulación de cada una de las fases del proceso de aprendizaje como son: reconocimiento, profundización y transferencia, logrando una coherencia metodológica con la guía de actividades propuesta.

Las unidades didácticas que lo conforman son tres, equivalentes al número de créditos asignados al curso académico. La primera y última unidad didáctica poseen cuatro capítulos cada una, y la segunda unidad consta de cinco capítulos.

La primera unidad, Teoría de los sistemas operativos, está orientado a acercar al interesado en los conceptos básicos y definición de lo que es un sistema operativo. En esta unidad se desarrollan capítulos como: Principios de los sistemas operativos, Características de los sistemas operativos, Interbloqueo y arquitectura cliente/servidor. Todos ellos para introducirnos al mundo de los sistemas operacionales, su historia, evolución, clasificación, estructura e infraestructura de aplicación.

Ya revisadas las bases teóricas y funciones principales, estamos listos para continuar el tema de cómo gestionan y administran los diferentes recursos del sistema computacional, tema que le corresponde a la Unidad Didáctica 2.

Administración de Recursos. Dichos recursos los aborda el curso así: En el Capítulo 1. La administración de los procesos, en el capítulo 2. Administración de la memoria, en el tercero Administración de dispositivos, en el cuarto Administración de archivos; y por último y como parte fundamental de un SO está la protección y seguridad.

Con esta temática terminamos de conocer a fondo cómo es un SO, cómo está organizado interna y externamente, para así poder brindar el adecuado soporte a cualquier sistema monosusuario o multisusuario que se nos presente.

Ahora queda solamente aplicar toda la fundamentación teórica desarrollada en las dos unidades iniciales a los principales sistemas operativos del mercado. De esto se encarga la unidad didáctica 3. Principales sistemas operativos, que realiza una clasificación de la siguiente manera: En primer lugar los sistemas operativos de la familia Windows, en segundo lugar el sistema operativo UNIX/LINUX y en la agrupación Otros sistemas operativos se habla de sistemas como novell, OS/2 y VMS.

Por último, finaliza el módulo con el capítulo concerniente a Arquitectura de las comunicaciones, considerándolo básico para acercarnos, de manera muy general, a las diferentes arquitecturas de red en las que se deben instalar y manejar los SO.

Este módulo fue desarrollado tomando como referencia documentación y estudios realizados de los diferentes sistemas operativos, en cada uno de los capítulos, se relacionan las fuentes bibliográficas específicas sobre las cuales se trabajó. Además es importante recordar, que este módulo debe ir articulado con las diferentes actividades planteadas en la guía didáctica, pues es un curso metodológico, es decir tiene un componente de aplicación y trabajo en grupo.

Por último, y como siempre, recomiendo que para facilitar el aprendizaje es importante consultar la bibliografía descrita, utilizar la biblioteca virtual y el acceso a Internet, con esto se está potenciando la capacidad de investigación y de auto gestión para llegar al conocimiento, según sean los logros y/o debilidades encontradas en cada uno de los pasos del proceso a seguir.

Recuerden que el éxito del proceso sólo depende de cada uno, de sus intereses y de sus necesidades.

OBJETIVOS

1. Fundamentar, desde un principio, la estructura, funcionamiento y administración de recursos de los sistemas operativos, como base para el análisis y diseño de sistemas de comunicación.
2. Relacionar los principios, estructuras, aplicación y tipos de sistemas operativos con las características y funcionamiento de algunos de los principales sistemas operativos.
3. Conocer e identificar de manera clara los conceptos, elementos, características, propiedades de los sistemas operativos y su relación con el campo de aplicación, teniendo en cuenta la integración de elementos tecnológicos y organizacionales.
4. Determinar y sustentar la aplicación de los sistemas operativos según las características, ventajas de instalación y administración de recursos.

UNIDAD DIDÁCTICA 1

TEORÍA DE LOS SISTEMAS OPERATIVOS

INTRODUCCIÓN

Aunque en este nivel de estudios, se supone que ya hemos visto, trabajado y explotado algunos de los sistemas operativos del mercado, muy seguramente no conocemos a fondo la teoría en la cual se basa su diseño y desarrollo. Y esto aunque a veces no lo reconocemos es fundamental a la hora de “aprovechar” al máximo los recursos de un sistema computacional cualquiera.

Aquí se explica qué son los sistemas operativos, qué hacen y cómo están diseñados y construidos. Se explica cómo se ha desarrollado el concepto de un sistema operativo, cuáles son sus características comunes y lo que hace el sistema operativo para el usuario y para el administrador del sistema de cómputo.

Esta unidad es apropiada para los estudiantes que se inician en esta materia y para aquellos que son “expertos” pero desean saber más acerca de los detalles internos de los mismos.

OBJETIVOS

1. Explorar los principios de los sistemas operativos, como conceptos, estructura historia y evolución.
2. Diferenciar con exactitud los diversos tipos de sistemas operativos, identificando claramente sus características funcionales y de diseño.
3. Reconocer el Interbloqueo, como el principal problema que debe solucionar cualquier sistema operativo, identificando sus características y alternativas de solución existentes.
4. Identificar la estructura física y organizacional de la arquitectura cliente servidor, como principal esquema de trabajo de los sistemas operativos.

CAPÍTULO 1. PRINCIPIOS DE LOS SISTEMAS OPERATIVOS

Actividad inicial:

Recordemos cuál es el concepto de sistema operativo. Y reflexionemos de forma individual con respecto a:

- Ha trabajado con algún sistema operativo?
- Con cuál o cuáles?
- Tiene un sistema operativo favorito? Diga cuál y porqué?
- Conoce algo acerca de la estructura interna de un sistema operativo?
- Conoce una estructura específica de un sistema operativo cualquiera?

Enseguida de esta reflexión aborde toda la temática y concluya.

1.1 Concepto de sistema operativo

Conceptuar el término “sistema operativo” (ó S.O como se nombra en algunas partes del módulo) no es simple, precisamente es el objetivo primordial del módulo. Existen diversas definiciones de lo que es un sistema operativo, pero no hay una definición exacta, es decir una que sea estándar; a continuación se presentan algunas:

- Un sistema operativo es un programa que actúa como intermediario entre el usuario y el hardware de un computador y su propósito es proporcionar un entorno en el cual el usuario pueda ejecutar programas.
- Un sistema operativo es el código que acompaña la ejecución de cualquier aplicación.
- Un sistema operativo es un programa que dirige y administra los recursos de un sistema computacional. Provee un conjunto de cualidades que facilitan el acceso de las aplicaciones a estos recursos, buscando siempre independencia del hardware.
- Un sistema operativo es la parte del sistema de cómputo que administra el hardware y el software.
- Un sistema operativo es un conjunto de programas que ordenadamente relacionados entre si, contribuyen a que el hardware de la computadora lleve a cabo su trabajo correctamente.
- Un sistema operativo es el soporte lógico que controla el funcionamiento del equipo físico.
- Un sistema operativo es el programa que oculta la verdad del hardware al programador y presenta una vista simple y agradable de los archivos nominados que pueden leerse y escribirse.

Una definición que llama mucho la atención y particularmente comparto es:

- *Un sistema operativo es un programa cuya estructura es lo suficientemente general para independizarse del hardware, pero la implementación debe ser lo suficientemente particular para aprovechar de forma eficiente el hardware.*

Existen definiciones más amplias y un poco más específicas, como:

- Se pueden imaginar un sistema operativo como los programas, instalados en el software o firmware, que hacen utilizable el hardware. El hardware proporciona la "capacidad bruta de cómputo"; los sistemas operativos ponen dicha capacidad de cómputo al alcance de los usuarios y administran cuidadosamente el hardware para lograr un buen rendimiento.
- Los sistemas operativos son ante todo administradores de recursos; el principal recurso que administran es el hardware del computador; además de los procesadores, los medios de almacenamiento, los dispositivos de entrada/salida, los dispositivos de comunicación y los datos
- El objetivo principal de un sistema operativo es, lograr que el sistema de computación se use de manera *cómoda*, y el objetivo secundario es que el hardware del computador se emplee de manera *eficiente*
- Un sistema operativo es un conjunto de programas que controla la ejecución de programas de aplicación y actúa como una interfaz entre el usuario y el hardware de una computadora, esto es, un sistema operativo explota y administra los recursos de hardware de la computadora con el objeto de proporcionar un conjunto de servicios a los usuarios del sistema.

En resumen, se podría decir que los sistemas operativos son un conjunto de programas que crean la interfaz del hardware con el usuario, y que tiene dos funciones primordiales, que son:

- **Gestionar el hardware:** Se refiere al hecho de administrar de una forma más eficiente los recursos de la máquina.
- **Facilitar el trabajo al usuario:** Permite una comunicación con los dispositivos de la máquina.

Si se analizan con detenimiento cada una de las anteriores definiciones, se puede concluir que en general un sistema operativo se describe desde dos puntos de vista:

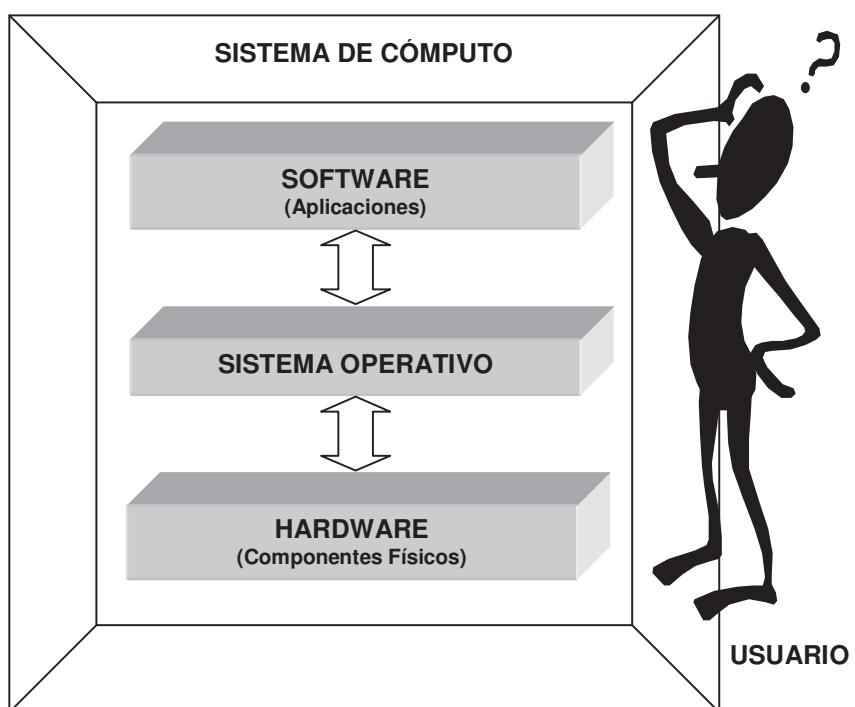
- **Usuario final** y,
- **Usuario administrador.**

Queda, entonces, revisar los conceptos dados y determinar a cuál punto de vista corresponde cada uno.

A lo largo del curso se va a trabajar la concepción de sistema operativo como:

Aquella herramienta lógica que proporciona al usuario un entorno amigable, permite interactuar y establecer una comunicación entre el hardware (componentes físicos) y el software (aplicaciones) de un sistema de cómputo.

Gráfica 1. Concepción de sistema operativo



1.2 Estructura de un sistema computacional

Un sistema operativo es una parte importante de cualquier sistema de computación.

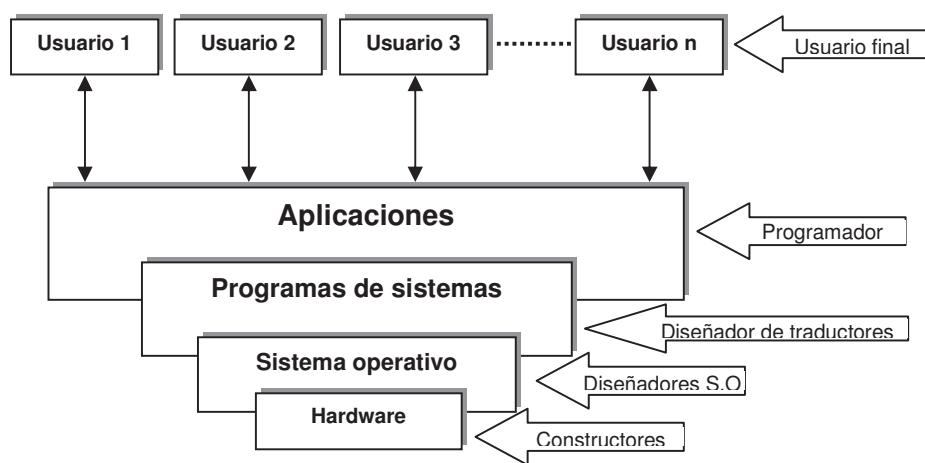
Un sistema computacional es una plataforma sobre la cual se construyen los llamados sistemas de información, hoy en día, necesarios para la administración exitosa de cualquier organización.

Un sistema de computación puede dividirse en cuatro componentes: el *hardware*, el *sistema operativo*, los *programas de aplicación* y los *usuarios*.

El hardware (Unidad Central de Procesamiento (CPU), memoria y dispositivos de entrada/salida (E/S)) proporcionan los recursos de computación básicos.

Los programas de aplicación definen la forma en que estos recursos se emplean para resolver los problemas de computación de los usuarios. Se pueden dividir en programas de sistemas como compiladores, assembler, editores, herramientas de monitoreo y mantenimiento; y en aplicaciones propiamente dichas como sistemas de bases de datos, juegos de video, programas para negocios, navegadores, etc.

Gráfica 2. Estructura de un sistema computacional



El sistema operativo se encuentra almacenado en la memoria secundaria. Primero se carga y ejecuta un pedazo de código que se encuentra en el procesador, el cual carga el BIOS, y este a su vez carga el sistema operativo que carga todos los programas de aplicación y software variado.

Antes de entrar a considerar los detalles de un S.O. se necesitan conocer los bloques de hardware que componen un sistema computacional.

Como la función principal de un S.O. es aliviar las tareas de E/S, revisaremos los dispositivos y estructura de la entrada y salida de un sistema computacional. También examinaremos los mecanismos de protección que provee la CPU para el S.O.

Veamos:

1.2.1. Operación de un sistema computacional

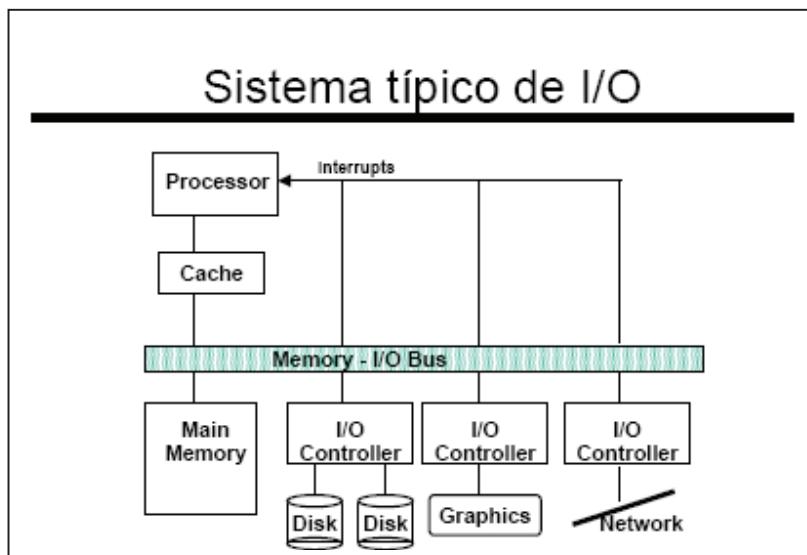
Un sistema computacional consiste en una CPU y un número de dispositivos que tienen acceso a una memoria común vía un bus de interconexión.

Los controladores de dispositivos se encargan de determinados dispositivos: discos, audio, mouse, módem, etc.

La CPU y los dispositivos pueden actuar en forma concurrente. Sólo compiten por el acceso a la memoria que es común.

El controlador de memoria provee un acceso sincronizado a la memoria RAM.

Gráfica 3. Sistema de entrada/salida E/S (Input-output I/O)¹



a. Partida

En la partida se necesita un programa inicial. Este programa se llama *bootstrap*. El bootstrap inicializa todo el sistema: registros de CPU, controladores de dispositivos, memoria, etc.

El primer proceso que ejecuta el S.O. Se llama init. Init espera la ocurrencia de algún evento.

¹ CAÑAS R, Javier (2002). Curso sistemas operativos (Formato .pdf). Capítulo 2, p. 3

b. Interrupciones

Una *interrupción* es un evento que altera la secuencia en que el procesador ejecuta las instrucciones. La interrupción es generada por el hardware del sistema de cómputo.

El sistema de interrupciones es una parte muy importante de la arquitectura de un computador.

Los sistemas operativos modernos son sistemas que reaccionan ante interrupciones, es decir, si no hay E/S, ni procesos ejecutándose, el S.O. está quieto y atento a que ocurra alguna interrupción.

¿Qué ocurre con el S.O ante una interrupción?

- El sistema operativo toma el control (es decir, el hardware pasa el control al sistema operativo).
- El sistema operativo guarda el estado del proceso interrumpido. En muchos sistemas esta información se guarda en el bloque de control de proceso interrumpido.
- El sistema operativo analiza la interrupción y transfiere el control a la rutina apropiada para atenderla; en muchos sistemas actuales el hardware se encarga de esto automáticamente.
- La rutina del *manejador de interrupciones* procesa la interrupción.
- Se restablece el estado del proceso interrumpido (o del siguiente proceso).
- Se ejecuta el proceso interrumpido (o el siguiente proceso).

Una interrupción puede ser iniciada específicamente por un proceso en ejecución (en cuyo caso se suele denominar (trap), y se dice que está *sincronizada* con la operación del proceso) o puede ser causada por algún evento que puede estar relacionado o no con el proceso en ejecución (en cuyo caso se dice que es *asíncrona* con la operación del proceso).

Los sistemas orientados hacia las interrupciones pueden sobrecargarse. Si estas llegan con mucha frecuencia, el sistema no será capaz de atenderlas. En algunos sistemas orientados hacia el teclado, cada tecla presionada almacena en la memoria un código de un byte y genera una interrupción para informar a la CPU que un carácter está listo para ser procesado. Si la CPU no puede procesar el dato antes de que se presione la siguiente tecla, se pierde el primer carácter.

Clases de Interrupciones

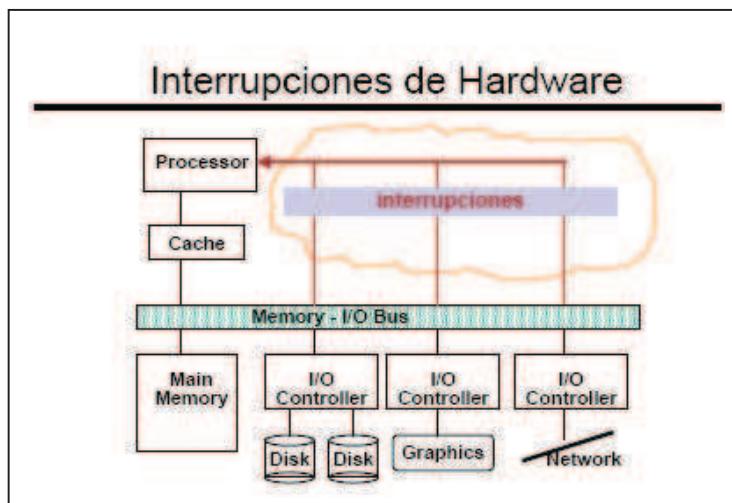
Existen seis clases de interrupciones:

- **Interrupciones SVC (supervisor call, llamadas al supervisor).** Son iniciadas por un proceso en ejecución que ejecute la *instrucción SVC*. Una

SVC es una petición generada por el usuario de un servicio particular del sistema, como realizar una operación de entrada/salida, obtener más memoria o comunicarse con el operador del sistema. El mecanismo de las SVC ayuda a proteger el sistema operativo de las acciones de los usuarios. Un usuario no puede entrar arbitrariamente al sistema operativo, sino que debe solicitar un servicio por medio de una SVC. El sistema operativo está al tanto de todos los usuarios que intentan rebasar sus límites y puede rechazar ciertas peticiones si el usuario no tiene los privilegios necesarios.

- **Interrupciones de E/S.** Son iniciadas por hardware de entrada y salida. Estas interrupciones indican a la CPU el cambio de estado de un canal o dispositivo. Las interrupciones de E/S se producen cuando finaliza una operación de E/S o cuando un dispositivo pasa al estado listo.
- **Interrupciones externas.** Son causadas por diversos eventos, incluyendo la expiración de un cuantum de un reloj que interrumpe, la pulsación de la tecla de *interrupción* de la consola o la recepción de una señal procedente de otro procesador en un sistema de múltiples procesadores.
- **Interrupciones de reinicio.** Se produce cuando se presiona el botón de reinicio de la PC o cuando llega de otro procesador una instrucción de reinicio en un sistema de multiprocesamiento.
- **Interrupciones de verificación del programa.** Son causadas por una amplia clase de problemas que pueden ocurrir cuando se ejecutan las instrucciones en lenguaje de máquina de un programa. Dichos problemas incluyen la división entre cero, el exceso o defecto de los números que pueden ser manejados por las operaciones aritméticas, el intento de hacer referencia a una localidad de memoria que esté fuera de los límites de la memoria real. Muchos sistemas ofrecen a los usuarios la opción de especificar las rutinas que deben ejecutarse cuando ocurra una interrupción de verificación del programa.
- **Interrupciones de verificación de la máquina.** Son ocasionadas por el mal funcionamiento del hardware.

Gráfica 4. Interrupciones de Hardware²



1.2.2 Estructura del sistema de E/S

Para iniciar una operación de E/S, el S.O. carga registros apropiados de los controladores de dispositivos. El controlador examina el registro, inicia la operación e informa de su término a la CPU mediante una interrupción

La E/S puede ser sincrónica o asincrónica respecto al proceso que la inicia.

a. E/S sincrónica y asincrónica

➤ **Sincrónica:**

- Se inicia la operación de E/S.
- Al finalizar la transferencia el control vuelve al proceso usuario.

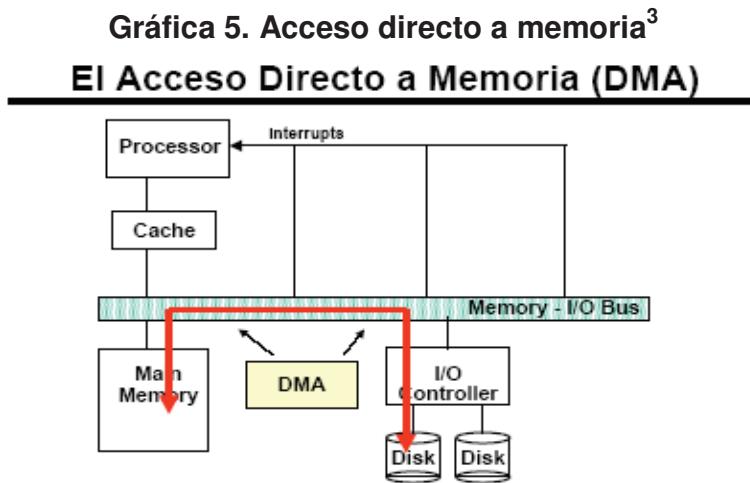
➤ **Asincrónica:**

- El control vuelve al proceso usuario sin necesidad de esperar el término de la transferencia.

b. El acceso directo a memoria (DMA)

Una forma de mejorar el desempeño de un sistema computacional es liberar a la CPU del control de la transferencia del Buffer del controlador a la memoria principal. De esta forma se genera sólo una interrupción por bloque en vez de una interrupción por Byte

² Ibid. Capítulo 2. p. 4



c. La memoria principal

La memoria principal (RAM) y el archivo de registro son el único almacenamiento que la CPU puede accesar directamente.

¿Cómo accesar mediante un programa los dispositivos?

Muchos computadores proveen memory-mapped i/o (dispositivos mapeados en memoria principal).

d. memory-mapped i/o

Esta modalidad considera a los registros de los dispositivos mapeados en determinadas direcciones de la memoria principal. El programa sólo debe hacer referencia a determinadas direcciones de memoria para iniciar transferencias.

Por ejemplo en los PC, cada punto de la pantalla de video está mapeada a una determinada dirección de memoria.

e. Discos

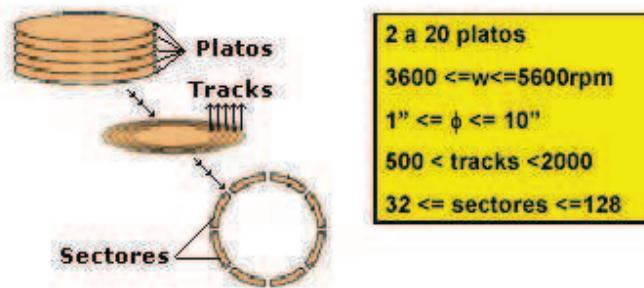
- Los discos permiten almacenamiento masivo.
- Las velocidades de rotación varían entre 60 a 150 Hz.
- Los tiempos involucrados en una transferencia son:

- Tiempo de transferencia (velocidad angular)
- Tiempo rotacional
- Tiempo de seek

³ Ibid. Capítulo 2. p. 8

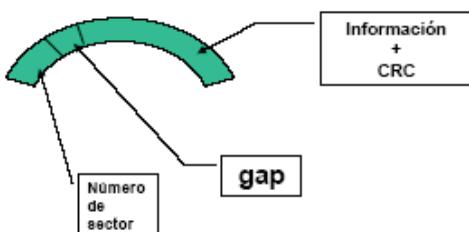
Gráfica 6. Organización de un disco⁴

Organización de un Disco



Gráfica 7. Información de un sector⁵

Información de un sector



Normalmente el número de sectores por track es el mismo
Otra opción es grabar con densidad constante (SCSI)

⁴ Ibid. Capítulo 2, p. 10

⁵ Ibid. Capítulo 2, p. 10

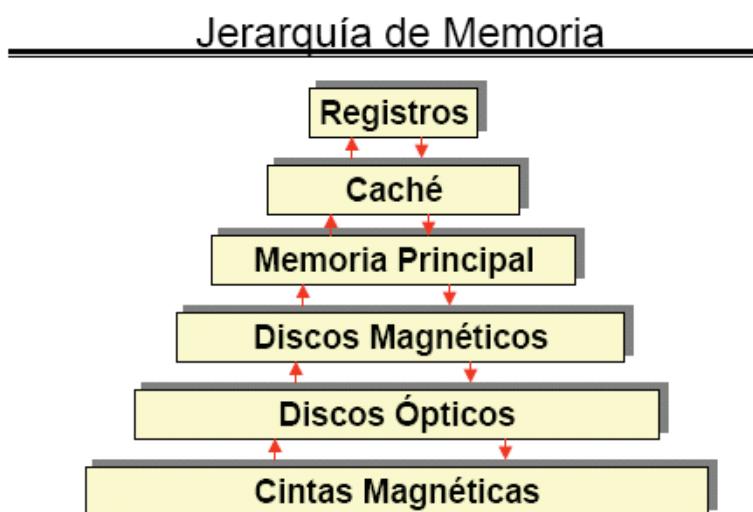
Gráfica 8. El cilindro⁶

f. Jerarquía de memoria

En un sistema computacional existe una gran variedad de almacenamiento. El almacenamiento está organizado jerárquicamente.

La jerarquía de almacenamiento responde al compromiso entre velocidad y costo de almacenamiento: el costo de almacenar un bit en una memoria muy rápida es caro.

Otro aspecto a considerar es la *volatilidad*.

Gráfica 9. Jerarquía de memoria⁷

⁶ Ibid. Capítulo 2, p. 11

⁷ Ibid. Capítulo 2, p. 12

g. La memoria caché

La memoria caché es una parte muy importante de un sistema computacional.

La información se mantiene en algún sistema de almacenamiento y en la medida que se usa es copiada en una memoria más rápida temporalmente.

Cuando se necesita una información particular, primero se verifica si está en la caché. Si está se usa directamente y si no está se extrae del medio de almacenamiento y se copia en la caché.

Como su tamaño es limitado resulta muy importante su administración.

1.2.3 Protecciones de hardware

Los primeros computadores eran sistemas monousuarios.

En la medida que los S.O. evolucionaron fue necesario compartir recursos para mejorar la eficiencia del sistema.

El compartir mejora la eficiencia y aumenta los problemas:

- Sistema multiprogramado ante condiciones de error.
- Compartir dispositivos.

a. Modo Dual

Para asegurar una correcta operación se debe proteger al S.O. y los programas frente a situaciones de error.

La protección se requiere para cualquier recurso compartido.

El hardware provee una importante protección llamada Modo Dual

Se agrega un bit al hardware llamado *bit de modo* para indicar dos modos posibles de operación.

Los modos de operación son dos:

- *Modo Monitor* (también llamado kernel o modo sistema)
- *Modo usuario*

Modo Monitor: este es el modo en el cual el S.O. toma el control del computador. Sólo en este modo se pueden ejecutar instrucciones llamadas privilegiadas y accesar estructuras de datos internas del S.O.

Modo Usuario: modo normal para código usuario.

La falta de apoyo de hardware de protección trae serios problemas en los S.O. Un ejemplo es el S.O. originalmente escrito para el Intel 8080 que no tiene bit de modo:

- Cualquiera puede sobre escribir el S.O.
- Muchos programas pueden hacer E/S al mismo tiempo.

A partir del 80486 se incorporó el bit de modo y así fue posible soportar S.O. Como Windows NT, Windows 2000, Windows 2003 Server, OS/2 y Linux.

b. Protección de E/S

Para prevenir que un usuario realice una operación ilegal de E/S se definen instrucciones privilegiadas.

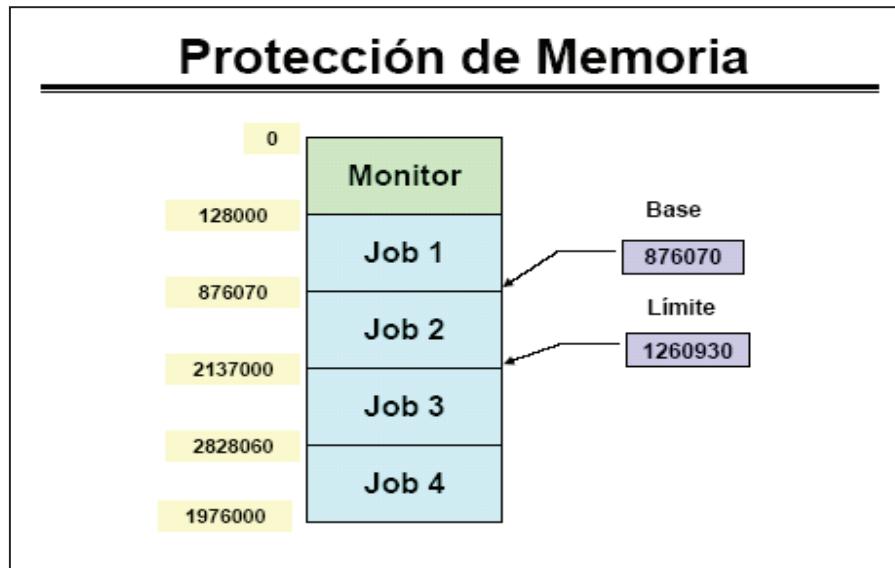
De esta forma un usuario no puede ejecutar instrucciones de E/S directamente. Lo debe hacer a través del S.O.

Nunca un programa usuario debe tener el control del sistema bajo modo monitor.

¿Qué pasaría si se tiene acceso al vector de interrupción? Se obtiene el control en modo monitor.

c. Protección de memoria

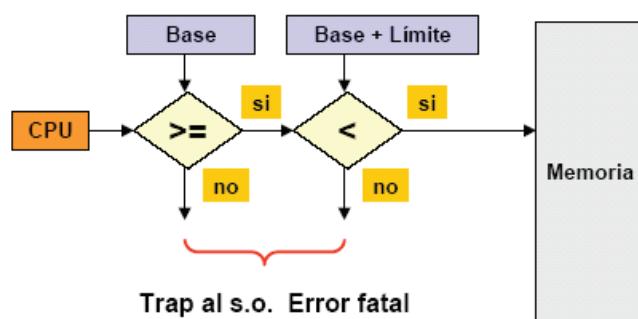
- Se debe proteger el vector de interrupción de ser modificado por programas de usuario.
- Se debe proteger la alteración de rutinas de servicio de interrupción.
- Se debe proteger también un programa usuario de otro programa usuario.
- Cada espacio debe estar protegido.
- El hardware provee dos registros:
 - Registro base: Menor dirección legal
 - Registro límite: Tamaño del espacio protegido.

Gráfica 10. Protección de memoria⁸

- Cada dirección generada en modo usuario es comparada con los registros Base y Límite.
- Cada intento por violar una región protegida genera una interrupción al S.O., el cual lo trata como un error fatal.
- Por supuesto que sólo el S.O. puede cambiar los contenidos de los registros Base y Límite (se cambian en modo monitor)

Gráfica 11. Control de la protección de memoria⁹

Protección de Memoria

⁸ Ibid. Capítulo 2, p. 16⁹ Ibid. Capítulo 2, p. 17

Por qué se estudian los sistemas operativos?

- Los sistemas operativos son sistemas de software complejos. El entendimiento de los conceptos utilizados y la implementación de estos programas proporcionan desafíos y ejemplos para cualquier persona.
- El conocimiento de los sistemas operativos permite realizar aplicaciones que aprovechen los recursos eficientemente.
- Para el diseño de sistemas operativos.
- Para la creación de sistemas operativos personales con base en los sistemas operativos existentes. (Personalización de versiones)

1.3 Historia de los sistemas operativos

Para tratar de comprender los requisitos de un S.O y el significado de sus principales características, es útil considerar como han ido evolucionando éstos con el tiempo.

Existen diferentes enfoques o versiones de cómo han ido evolucionando los sistemas operativos.

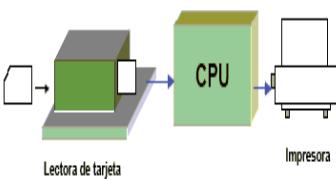
La primera de estas versiones podría ser esta:

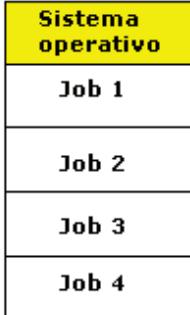
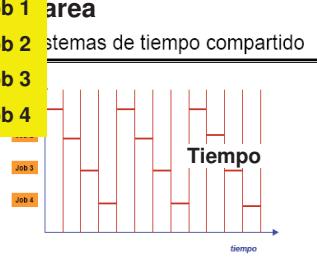
Periodo	Avance
Años 40's	Se introducen los programas bit a bit, por medio de interruptores mecánicos y después se introdujo el lenguaje de máquina que trabajaba por tarjetas perforadas.
Finales de los años 40 hasta la mitad de los años 50's	Con las primeras computadoras, el programador interactuaba de manera directa con el hardware de la computadora, no existía realmente un sistema operativo; las primeras computadoras utilizaban bulbos, la entrada de datos y los programas se realizaban a través del lenguaje máquina (bits) o a través de interruptores.
Durante los años 50's y 60's	A principio de los 50's, la compañía General's Motors implantó el primer sistema operativo para su IBM 170. Empiezan a surgir las tarjetas perforadas las cuales permiten que los usuarios (que en ese tiempo eran programadores, diseñadores, capturistas, etc.), se encarguen de modificar sus programas. Establecían o apartaban tiempo, metían o introducían sus programas, corregían y depuraban sus programas en su tiempo. A esto se le llamaba trabajo en serie. Todo esto se traducía en pérdida de tiempo y tiempos de programas excesivos.
En los años 60's y 70's	Se genera el circuito integrado, se organizan los trabajos y se generan los procesos Batch (por lotes), lo cual consiste en determinar los trabajos comunes y realizarlos todos juntos de una sola vez. En esta época surgen las unidades de cinta y el cargador de programas, el cual se considera como el primer tipo de sistema operativo.

En los 80's	Inició el auge de la INTERNET en los Estados Unidos de América. A finales de los años 80's comienza el gran auge y evolución de los sistemas operativos. Se descubre el concepto de multiprogramación que consiste en tener cargados en memoria a varios trabajos al mismo tiempo, tema principal de los sistemas operativos actuales.
Los 90's y el futuro	Entramos a la era de la computación distribuida y del multiprocesamiento a través de múltiples redes de computadoras, aprovechando el ciclo del procesador. Se tendrá una configuración dinámica con un reconocimiento inmediato de dispositivos y software que se añada o elimine de las redes a través de procesos de registro y localizadores. La conectividad se facilita gracias a estándares y protocolos de sistemas abiertos establecidos por organizaciones como la Organización Internacional de estándares (ISO-International Standard Organization), fundación de software abierto, todo estará mas controlado por los protocolos de comunicación OSI y por la red de servicios digital ISDN.

Se ha desarrollado otra versión, la cual se ha hecho con base a la evolución del hardware:

- Sistemas Batch simples
- Sistemas Batch multiprogramados
- Sistemas de tiempo compartido
- Computadores personales
- Sistemas paralelos
- Sistemas distribuidos
- Sistemas de tiempo real

Tipo de sistema	Descripción
Sistemas Batch Simples	 <p>Los primeros computadores eran grandes máquinas que se operaban desde una consola. La entrada y salida se hacía usando tarjetas perforadas y cinta magnética. La interacción de un usuario con el sistema computacional no era directa: se preparaba un job que consistía en un conjunto de tarjetas: programa, datos y tarjetas de control.</p> <p>El S.O:</p> <ul style="list-style-type: none"> ➤ Tenía una función muy simple: transferir el control entre una tarea (job) y la siguiente. ➤ Residía completamente en memoria. <p>Para hacer más eficiente el trabajo, los operadores agrupaban tareas en tandas o lotes (batch).</p> <p>La característica más importante es la falta de interacción entre el usuario y el sistema durante la ejecución.</p> <p>Los job se preparan y entregan al sistema y después de un tiempo se entrega el resultado vía una lista o impresión.</p> <p>La CPU pasa desocupada la mayor parte del tiempo:</p> <ul style="list-style-type: none"> ➤ La velocidad de los elementos mecánicos: impresora y lectora es mucho más baja que la CPU. <p>¿Cómo solucionar este problema?</p> <ul style="list-style-type: none"> - Tecnología de disco (Spooling) – Ver explicación al final del cuadro

Tipo de sistema	Descripción
Sistemas Batch Multiprogramados 	<p>El Spooling mantiene una estructura de datos con todos los jobs listos para ser ejecutados en un área de disco.</p> <p>Esta estructura permite seleccionar cualquier job del conjunto. Con esta estructura es posible mejorar la utilización de la CPU.</p> <p>La selección de un job para su ejecución de un conjunto se denomina <i>itineración</i> de job (scheduling).</p> <p>La itineración de jobs permite la multiprogramación.</p> <p>La multiprogramación aumenta la utilización de la CPU al organizar los jobs de manera tal que la CPU siempre tenga algún job que ejecutar.</p> <p>Para esto se mantienen los jobs en memoria principal. El S.O. selecciona un job, lo ejecuta y cuando el job debe esperar por E/S, se selecciona otro job.</p> <p>Cuando un job necesita esperar por algún dispositivo, el sistema operativo conmuta de un job a otro. Cuando la transferencia del dispositivo termina, se vuelve al job nuevamente</p>
Sistemas de tiempo compartido – Time sharing 	<p>Los sistemas batch multiprogramados permiten usar recursos eficientemente, pero los usuarios no pueden interactuar con sus aplicaciones.</p> <p>El tiempo compartido (time sharing) o multitarea es una extensión de la multiprogramación. La CPU ejecuta múltiples jobs, pero la conmutación de un job a otro ocurre con una frecuencia tal que los usuarios piensan que interactúan con el programa mientras éste corre.</p> <p>Los primeros sistemas Batch eran completamente interactivos. El usuario tomaba el control completo del sistema a través de la consola.</p> <p>Los sistemas de tiempo compartido se desarrollaron para proporcionar el uso interactivo de un computador a costo razonable.</p> <p>Cada usuario tiene al menos un programa en memoria.</p> <p>Un programa que se carga y ejecuta se denomina proceso. Cuando un proceso se ejecuta, lo hace por un tiempo corto antes que termine o necesite E/S.</p> <p>La entrada y salida (E/S) también puede ser interactiva.</p> <p>Los sistemas operativos de tiempo compartido son más complejos que los sistemas batch multiprogramados. Entre otras cosas se requiere protección especial de áreas de memoria.</p>
Sistemas PC – Computadores personales	<p>Los Computadores personales aparecieron en el mercado en la década del 70. El objetivo de los sistemas operativos de PC no es mejorar la eficiencia sino su amistosidad con el usuario.</p> <p>Ejemplos son: MS-DOS, MS Windows, Apple Macintosh, OS/2</p> <p>La tendencia es traspasar funcionalidades de grandes computadores a PC. Por ejemplo sistemas de protección de archivos, memoria virtual etc..</p> <p>Un buen ejemplo es el sistema MULTICS desarrollado en el MIT entre 1965 y 1970. Las ideas de MULTICS se tomaron en Bell Labs para desarrollar UNIX en 1970 para computadores DEC PDP-11. En los años 80, surgieron muchos sistemas tipo UNIX: W/NT, OS/2 , MAC OS y Windows XP y LINUX recientemente.</p>
Sistemas paralelos	<p>La mayoría de los sistemas computacionales actuales utiliza una sola CPU, sin embargo hay una tendencia hacia sistemas</p>

Tipo de sistema	Descripción
	<p>multiprocesadores.</p> <p>¿Qué se logra con multiprocesadores?</p> <p>Mayor desempeño (throughput): más trabajo por unidad de tiempo.</p> <p>Aceleramiento de tareas: cuando varios procesadores cooperan en la realización de una tarea, disminuye el tiempo de ejecución. Sin embargo la mejora de desempeño no es lineal respecto al número de procesadores por el tiempo de comunicación.</p> <p>¿Qué es mejor, un sistema de multiprocesamiento o varios sistemas simples?</p> <p>Es más económico un sistema de multiprocesadores (un disco, gabinete, fuentes de poder etc.)</p> <p>Mejora la confiabilidad: si las funciones se distribuyen inteligentemente, la caída de un procesador puede ser asumida por otro.</p>
Sistemas distribuidos	<p>La tendencia actual es distribuir la computación entre varios procesadores.</p> <p>Cada procesador tiene su memoria local. Los procesadores se comunican por líneas de comunicación, redes de alta velocidad o buses apropiados.</p> <p>Los procesadores de un sistema distribuido varían en tamaño y función: microprocesadores, minicomputadores, estaciones de trabajo y grandes sistemas computacionales.</p>
Sistemas de tiempo real	<p>Los sistemas operativos de Tiempo Real se usan cuando existen rígidos requerimientos de tiempo. Ejemplos:</p> <ul style="list-style-type: none"> - Sistemas de Control Industrial - Monitoreo médico - Control de encendido de motores - Sistemas de defensa <p>El procesamiento se debe hacer con restricciones de tiempo. En caso contrario se producen fallas.</p>

Algunos de los anteriores sistemas se revisarán en el capítulo siguiente: Características de Sistemas Operativos – Tipos de sistemas operativos.

Una tecnología que apareció con los sistemas de batch es: **Spooling**.

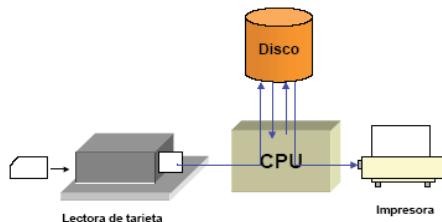
Spooling – Tecnología de discos

La introducción de la tecnología de discos entrega una solución: en vez de leer del disco y escribir a la impresora, la CPU interactúa directamente con el disco.

Si la CPU necesita leer una tarjeta, lee desde un área del disco. En forma similar cuando necesita imprimir una línea, la escribe en el disco

Esta forma de procesamiento se denomina Spooling (Simultaneous peripheral operation on-line)

Gráfica 12. Tecnología de discos¹⁰
Spooling



Observaciones

- El Spooling traslape la entrada y salida de un job con los cálculos (CPU) de otro job.
- Tiene un efecto directo e importante en el desempeño ya que mantienen a la CPU y los dispositivos trabajando a su máxima velocidad.

1.4 Conceptos de sistemas operativos

1.4.1 Llamadas al sistema

Hemos visto, por ejemplo, que las instrucciones de E/S son privilegiadas y por lo tanto sólo las puede ejecutar el S.O.

¿Cómo ejecutar e/s? Solicitando al S.O. a través de una *llamada al sistema*.

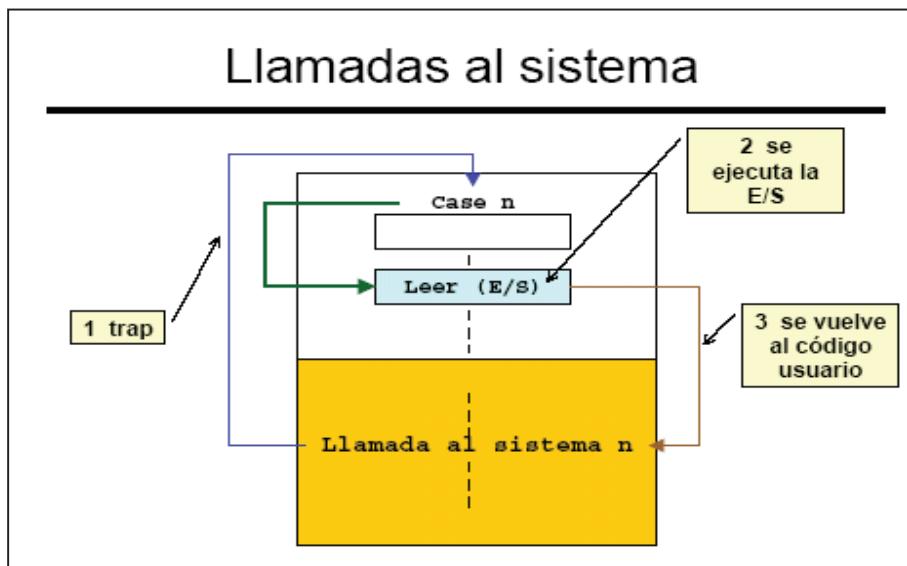
Las llamadas al sistema se usan en general para solicitar cualquier servicio al S.O. (Se tratan en este capítulo)

Una llamada al sistema genera una interrupción (trap) a una dirección específica del vector de interrupciones.

Cuando se ejecuta una llamada al sistema, el hardware la trata como una interrupción (excepción) de software. Los parámetros se pasan vía registro o direcciones de memoria.

La siguiente gráfica muestra el flujo de una llamada al sistema.

¹⁰ Ibid. Capítulo 1. p. 8

Gráfica 13. Flujo de una llamada al sistema¹¹

La interfaz entre el sistema operativo y los programas del usuario se define por medio del conjunto de "instrucciones extendidas" que el sistema operativo proporciona. Estas instrucciones extendidas son las **llamadas al sistema**. Las llamadas al sistema varían de un sistema operativo a otro (aunque los conceptos fundamentales tienden a ser análogos).

Las llamadas al sistema se clasifican normalmente en dos categorías generales: **aquellas que se relacionan con procesos** y las que **lo hacen con el sistema de archivo**

- **Por procesos:** Un proceso es básicamente un programa en ejecución. Consta del programa ejecutable y la pila o stack del programa, su contador de programa, apuntador de pila y otros registros, así como la otra información que se necesita para ejecutar el programa. En si el proceso en el concepto de los sistemas operativos es como el sistema de tiempo compartido. Esto es, que en forma periódica, el sistema operativo decide suspender la ejecución de un proceso y dar inicio a la ejecución de otro, por ejemplo, porque el primero haya tomado ya más de su parte del tiempo de la CPU, en terrenos del segundo.

Cuando un proceso se suspende temporalmente, debe reiniciarse después exactamente en el mismo estado en que se encontraba cuando se detuvo. Esto significa que toda la información relativa al proceso debe guardarse en forma explícita en algún lugar durante la suspensión. En muchos sistemas operativos, toda la información referente a cada proceso, diferente del contenido de su

¹¹ Ibid. Capítulo 2. p. 19

espacio de direcciones, se almacena en una tabla de sistema operativo llamada tabla de procesos, la cual es un arreglo (lista enlazada) de estructuras, una para cada proceso en existencia.

Por lo tanto, un proceso (suspendido) consta de su espacio de direcciones, generalmente denominado imagen del núcleo (en honor de las memorias de imagen de núcleo magnético que se utilizaron en tiempos antiguos) y su registro de la tabla de procesos, que contiene sus registros entre otras cosas.

- **Por sistema de archivo:** Una función importante del S.O. consiste en ocultar las peculiaridades de los discos y otros dispositivos de E/S y presentar al programador un modelo abstracto, limpio y agradable de archivos independientes del dispositivo. Las llamadas al sistema se necesitan con claridad para crear archivos, eliminarlos, leerlos y escribirlos. Antes de que se pueda leer un archivo, éste debe abrirse y después de que se haya leído debe cerrarse, de modo que las llamadas se dan para hacer estas cosas.

Antes de que un archivo pueda leerse o escribirse, éste debe abrirse, en cuyo instante se verifican los permisos. Si se permite el acceso, el sistema produce un entero pequeño llamado descriptor del archivo para utilizarse en operaciones subsiguientes. Si se prohíbe el acceso, se produce un código de error.

1.4.2 Shell (intérprete de comandos)

El sistema operativo es el código que realiza las llamadas al sistema. Los editores, compiladores, ensambladores, enlazadores e intérpretes de comandos definitivamente no son parte del sistema operativo, aunque son importantes y útiles. El Shell es el intérprete de comandos, a pesar de no ser parte del sistema operativo (está íntimamente ligado con este), hace un uso intenso de muchas características del sistema operativo y por tanto sirve como un buen ejemplo de la forma en que se pueden utilizar las llamadas al sistema. También es la interfaz primaria entre un usuario situado frente a su terminal y el sistema operativo.

Cuando algún usuario entra al sistema, un "shell" se inicia. El shell tiene la terminal como entrada y como salida estándar. Este da inicio al teclear solicitud de entrada, carácter como un signo de pesos, el cual indica al usuario que el shell está esperando un comando. En MS-DOS normalmente aparece la letra de la unidad, seguida por dos puntos (:), el nombre del directorio en que se encuentra y por último el signo de "mayor que" (>). Esto es: C:\>.

Las versiones gráficas de Windows tienen la opción de Ejecutar, el cual es el shell del sistema, normalmente en Inicio – Opción ejecutar.

En Linux se puede trabajar, la consola de comandos, como el intérprete de comandos **shell**.

1.4.3 Procesos

Uno de los conceptos más importantes que gira entorno a un sistema operativo es el de proceso. Un **proceso** es un programa en ejecución junto con el entorno asociado (registros, variables ,etc.).

El corazón de un sistema operativo es el **núcleo**, un programa de control que reacciona ante cualquier interrupción de eventos externos y que da servicio a los procesos, creándolos, terminándolos y respondiendo a cualquier petición de servicio por parte de los mismos.

Un proceso es una actividad que se apoya en datos, recursos, un estado en cada momento y un programa.

Cada proceso contiene, entre otros:

- Mapeo en memoria: Dónde está almacenado el .text, .data y el stack del proceso.
- El estado de registros.
- Tabla de archivos en uso: Estado de cada archivo
- Credenciales (UID, GID, EUID, GUID). Identificadores de usuarios.
- Otros (PID, PPID, contadores, estados, prioridades). Identificadores de procesos.

Los procesos pueden crear nuevos procesos, y heredar algunos atributos de su padre.

El SO provee medios de comunicación entre procesos. El proceso se comunica con el SO mediante las llamadas al sistema (syscalls), para, por ejemplo:

- Abrir un archivo
- Alocar mas memoria
- Crear un nuevo archivo
- Sobreescribir su .text

Otro medio de comunicación son las **señales**. (Análogas a las interrupciones, pero a nivel software). Un programa puede mandar señales a otros programas, el SO puede mandar señales al programa.

1.4.4 Archivos

Un **Archivo** es una unidad lógica de almacenamiento. Es una abstracción sobre el dispositivo físico (disco rígido, floppy, etc).

Conjunto de información relacionada guardada en un dispositivo secundario. Está asociado a dispositivos de almacenamiento no volátiles.

Para el usuario es un concepto de unidad de almacenamiento permanente, organizada bajo un esquema jerárquico de directorios, que le permite tener un orden lógico y control sobre su información.

Los atributos normales de un archivo son: nombre, tipo, tamaño, tiempos y credenciales.

El concepto de archivos y directorios se encuentra en prácticamente todos los sistemas operativos. La organización de los archivos posee un sistema jerárquico. Cada archivo se identifica con su directorio y nombre.

Existe un directorio padre de todos los directorios. Las referencias a archivos pueden ser:

- **Absolutas:** Referidas desde el directorio padre ó,
- **Relativas:** Referidas al directorio actual.

Cada archivo posee sus respectivos permisos y niveles de seguridad asignados.

1.4.5 Definiciones

- a. **Deadlocks (Abrazos mortales):** Cuando dos o más procesos se están esperando mutuamente, en una situación sin salida.
- b. **Memory Management (Manejo de Memoria):** Estrategia de determinado S.O para el uso de memoria. Los problemas a resolver son protección y mapeo de programas. Ej. Memoria Virtual.

Observaciones

- No existe una definición precisa de un S.O.
- Los S.O. existen porque existe la necesidad de resolver problemas usando sistemas computacionales.
- Tampoco existe una definición única sobre qué contiene y qué no contiene un S.O.
- Una definición práctica es que un S.O. es el código que siempre se ejecuta y acompaña la ejecución de las aplicaciones.

Ahora analice lo que Usted contestó en la actividad inicial sobre su conocimiento acerca de sistemas operativos, y después de abordado el tema vuelva a realizarse las mismas preguntas y compare sus respuestas: la inicial y la final. Concluya.

CAPÍTULO 2. CARACTERÍSTICAS DE LOS SISTEMAS OPERATIVOS

Actividad inicial:

Describa las características principales que a su juicio, tienen los sistemas operativos Windows y Linux (los que Usted normalmente maneja. Distribución y versión que disponga). Saque una lista, por cada sistema operativo, de ellas y explique el por qué lo considera así.

Así mismo saque una lista de los puntos que considera negativos de cada uno de los sistemas operativos que está analizando.

2.1 Funciones del sistema operativo

A continuación se muestran las funciones principales que realiza todo sistema operativo. Se puede decir que son las características del sistema operativo:

- **Conveniencia.** Un sistema operativo hace más conveniente el uso de una computadora.
- **Eficiencia.** Un sistema operativo permite que los recursos de la computadora se usen de la manera más eficiente posible.
- **Habilidad para evolucionar.** Un sistema operativo deberá construirse de manera que permita el desarrollo, prueba o introducción efectiva de nuevas funciones del sistema sin interferir con el servicio.
- **Encargado de administrar el hardware.** El sistema operativo se encarga de manejar de una mejor manera los recursos de la computadora en cuanto a hardware se refiere, esto es, asignar a cada proceso una parte del procesador para poder compartir los recursos.
- **Administración de dispositivos (gestionar a través del kernel).** Coordinando y manipulando los dispositivos conectados al ordenador. El sistema operativo se debe encargar de comunicar a los dispositivos periféricos, cuando el usuario así lo requiera. Además debe organizar los datos para acceso rápido y seguro.
- **Manejar las comunicaciones en red.** El sistema operativo permite al usuario manejar con alta facilidad todo lo referente a la instalación y uso de las redes de computadoras.
- **Procesamiento por bytes de flujo** a través del bus de datos.

- **Facilitar las entradas y salidas.** Un sistema operativo debe hacerle fácil al usuario el acceso y manejo de los dispositivos de Entrada/Salida de la computadora.

- **Técnicas de recuperación de errores.** Gestiona los errores de hardware y la pérdida de los datos.

- **Gestión de permisos y de usuarios.** Adjudica los permisos de acceso a los usuarios y evita que las acciones de uno afecten el trabajo que está realizando otro. El sistema operativo evita que los usuarios se bloqueen entre ellos, informándoles si esa aplicación está siendo ocupada por otro usuario.

- **Control de seguridad.** Debe proporcionar seguridad tanto para los usuarios como para el software y la información almacenada en los sistemas.

- **Control de concurrencia.** Establece prioridades cuando diferentes procesos solicitan el mismo recurso.

- **Administración de memoria.** Asigna memoria a los procesos y gestiona su uso.

- **Generación de estadísticas.**

- **Control de la ejecución de los programas.** Para ello, acepta los trabajos, administra la manera en que se realizan, les asigna los recursos y los conserva hasta su finalización.

- **Administración de periféricos.**

- **Permite que se puedan compartir el hardware y los datos entre los usuarios.**

El software de aplicación son programas que se utilizan para diseñar, tal como el procesador de palabras, lenguajes de programación, hojas de cálculo, etc.

El software de base sirve para interactuar el usuario con la máquina, son un conjunto de programas que facilitan el ambiente plataforma, y permite el diseño del mismo.

El Software de base está compuesto por:

- Cargadores.
- Compiladores.
- Ensambladores.
- Macros.

2.2 Tipos de sistemas operativos

Según la perspectiva con la que se observen los sistemas operativos, pueden realizarse múltiples clasificaciones. Entre ellas revisaremos las siguientes:

- ✓ Sistemas operativos por su estructura (visión interna)
- ✓ Sistemas operativos por los modos de explotación
- ✓ Sistemas operativos por los servicios que ofrecen y,
- ✓ Sistemas operativos por la forma en que ofrecen sus servicios (visión externa).

2.2.1 Sistemas operativos por su estructura – Visión interna¹²

Esta clasificación tiene en cuenta cómo se diseñan los sistemas a la hora de ser creados. Hay que tener en cuenta que, en la mayoría de los casos estas concepciones de diseño no se aplican aisladas, si no que puede haber interrelación entre ellas.

Se deben observar dos tipos de requisitos cuando se construye un sistema operativo, los cuales son:

Requisitos de usuario: Sistema fácil de usar y de aprender, seguro, rápido y adecuado al uso a que se le quiere destinar.

Requisitos del software: Donde se engloban aspectos como el mantenimiento, forma de operación, restricciones de uso, eficiencia, tolerancia frente a los errores y flexibilidad.

A continuación se describen las distintas estructuras que presentan los actuales sistemas operativos para satisfacer las necesidades que de ellos se quieren obtener.

a. Estructura monolítica

Es la estructura de los primeros sistemas operativos constituidos fundamentalmente por un solo programa compuesto de un conjunto de rutinas entrelazadas de tal forma que cada una puede llamar a cualquier otra. Las características fundamentales de este tipo de estructura son:

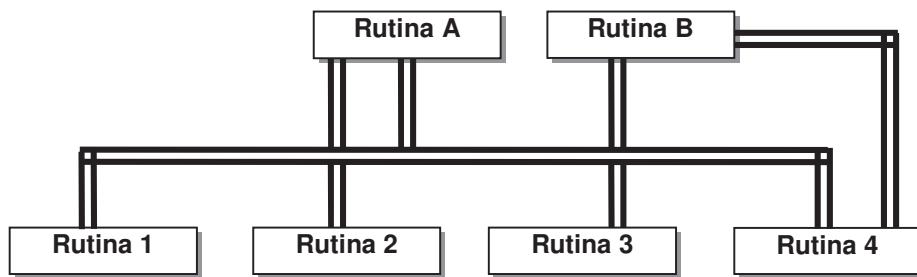
- Construcción del programa final a base de módulos compilados separadamente que se unen a través del ligador.

¹² ALCALDE, E. MORERA, J. PEREZ -CAMPANERO. (1992). *Introducción a los Sistemas Operativos*. Madrid, Mc Graw Hill. p. 33.

- Buena definición de parámetros de enlace entre las distintas rutinas existentes, que puede provocar mucho acoplamiento.
- Carecen de protecciones y privilegios al entrar a rutinas que manejan diferentes aspectos de los recursos de la computadora, como memoria, disco, etc.
- Generalmente están hechas a medida, por lo que son eficientes y rápidos en su ejecución y gestión, pero por lo mismo carecen de flexibilidad para soportar diferentes ambientes de trabajo o tipos de aplicaciones.

Es la estructura utilizada en los primeros sistemas operativos en la que todas las funciones se implementaban en el Kernel. Puede decirse que su estructura consiste en que no existe una estructura como tal.

Gráfica 14. Estructura monolítica



b. Estructura jerárquica – Por capas

A medida que los sistemas operativos fueron creciendo, fue siendo necesaria una mayor estructuración.

Se dividió el sistema operativo en pequeñas partes, de tal forma que cada una de ellas estuviera perfectamente definida y con una clara interface con el resto de elementos.

De acuerdo a las funciones principales del sistema operativo, vista en el numeral anterior (1.1.5 Funciones del sistema operativo), es posible analizar la estructura de un sistema operativo en cinco niveles. Los primeros dos niveles entrarían dentro de la parte del sistema operativo dependiente del hardware, el resto de los niveles pertenecen a la parte portable del mismo.

Cada uno de los niveles se comunica con el inmediatamente inferior y superior coordinando sus funciones.

- **Nivel 1: Gestión del procesador.** En este nivel se encuentra la parte del sistema operativo encargada de la gestión de la *CPU*. En los sistemas operativos multiproceso (es decir, que pueden ejecutar varios procesos a la

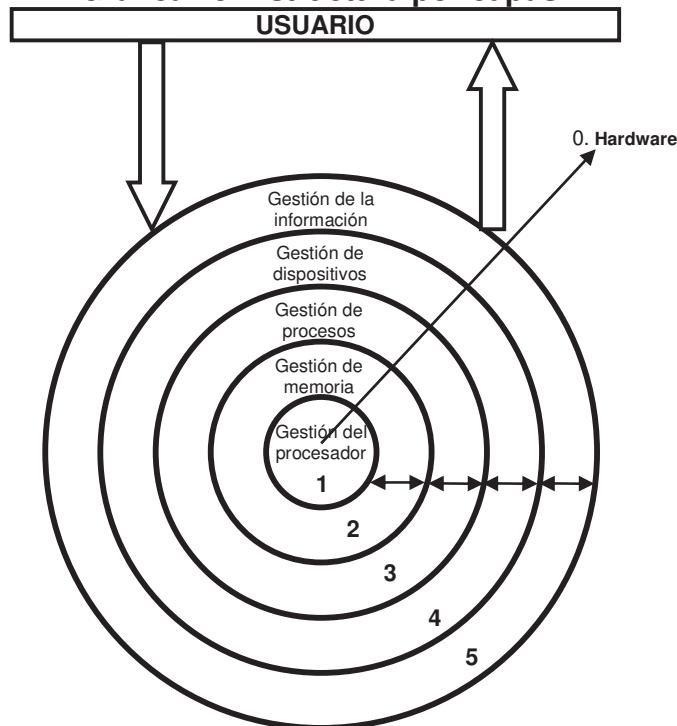
vez), este nivel se encarga de compartir la *CPU* entre los distintos procesos realizando funciones de sincronización, conmutación de la *CPU* y gestión de interrupciones.

- **Nivel 2: Gestión de memoria.** Este nivel es el encargado de repartir la memoria disponible entre los procesos. Se realizan funciones de asignación y liberación de memoria, y el control de violación de acceso a zonas de memoria no permitidas.
- **Nivel 3: Gestión de procesos.** Este nivel es el encargado de la creación y destrucción de los procesos, intercambio de mensajes y detección y arranque de los mismos.
- **Nivel 4: Gestión de dispositivos.** En este nivel se realiza la gestión de las entradas/salidas (*E/S*) en función de los dispositivos existentes. Entre otras, se encarga de las funciones de creación de procesos de *E/S*, asignación y liberación de dispositivos *E/S*, y planificación de la *E/S*.
- **Nivel 5: Gestión de la información.** El objetivo de este nivel es el de gestionar el espacio de nombres lógicos, utilizados para simplificar el acceso a los recursos, ya que mediante estos se sustituyen rutas de acceso que pueden ser muy largas y difíciles de recordar por un solo nombre, encargándose el sistema operativo, de forma totalmente transparente para el usuario, de realizar esta búsqueda de ruta. Otro de sus contenidos es la protección de la información realizando funciones de creación y destrucción de ficheros y directorios, apertura y cierre de ficheros, lectura y escritura de ficheros, y protección de acceso.

Es importante destacar que un mismo sistema operativo puede trabajar en múltiples plataformas hardware, por lo que debe adaptarse a las peculiaridades de cada una de ellas.

Una forma de representar esta estructura es mediante anillos concéntricos o “rings”. En el sistema de anillos, cada uno tiene una apertura, conocida como **trap** (o interrupción), 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. Se puede decir que las capas más internas son más privilegiadas que las externas.

En la base de la jerarquía se encuentra el hardware del computador, a veces denominado simplemente “máquina pura” o los “hierros desnudos”. En seguida se encuentran todos los anillos o capas propias del sistema operativo.

Gráfica 15. Estructura por capas.

c. Máquina virtual

Se trata de un tipo de sistemas operativos que presentan una interfaz a cada proceso, mostrando una máquina que parece idéntica a la máquina real subyacente.

Estos sistemas operativos separan dos conceptos que suelen estar unidos en el resto de sistemas: ***la multiprogramación y la máquina extendida***.

El objetivo de los sistemas operativos de máquina virtual es el de integrar distintos sistemas operativos dando la sensación de ser varias máquinas diferentes.

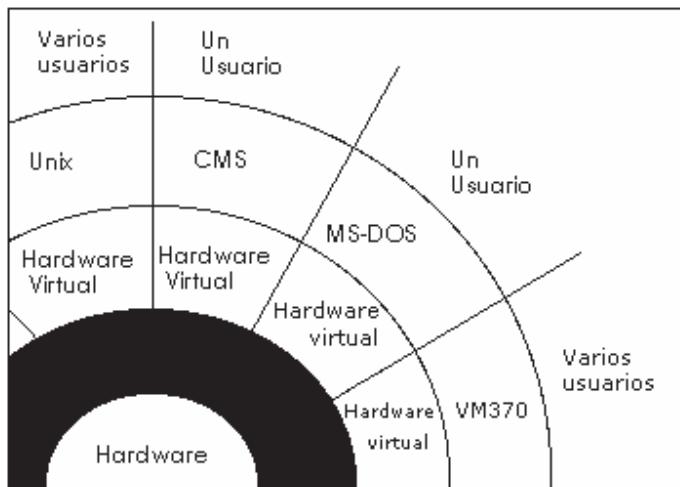
El núcleo de estos sistemas operativos se denomina ***monitor virtual*** y tiene como misión llevar a cabo la multiprogramación, presentando a los niveles superiores tantas máquinas virtuales como se soliciten. Estas máquinas virtuales no son máquinas extendidas, sino una réplica de la máquina real, de manera que en cada una de ellas se pueda ejecutar un sistema operativo diferente, que será el que ofrezca la máquina extendida al usuario.

La principal ventaja de esta estructura reside en que permite implementar varios tipos de sistemas operativos sobre cada máquina virtual.

La principal ventaja de esta estructura reside en que permite implementar varios tipos de sistemas operativos sobre cada máquina virtual. No obstante, presentan el problema de que los sistemas operativos implementados son disjuntos, lo cual complica enormemente la interacción, comunicación y compartición que necesitan los sistemas operativos actuales.

Un ejemplo de este tipo es el sistema VM370.

Gráfica 16. Máquina virtual



d. Cliente-servidor (Microkernel)

El tipo más reciente de sistemas operativos es el denominado **cliente-servidor**, que puede ser ejecutado en la mayoría de las computadoras, ya sean grandes o pequeñas.

Este sistema sirve para toda clase de aplicaciones, es de propósito general y cumple con las mismas actividades que los sistemas operativos convencionales: el núcleo y los procesos, presentando grandes diferencias en cuanto a la forma de distribuir los trabajos entre sus distintas partes.

Suministra mecanismos adecuados para la gestión de:

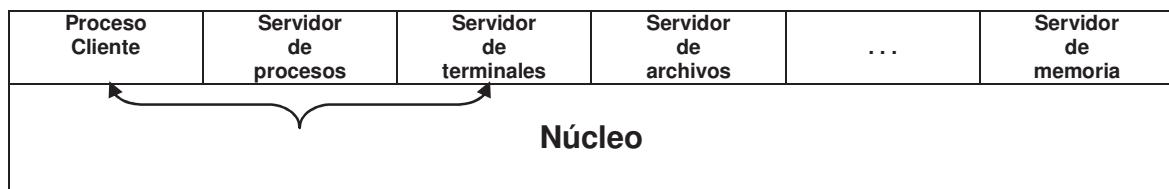
- Procesos.
- Memoria.
- Comunicación entre procesos.

El núcleo tiene como misión establecer la comunicación entre los clientes y los servidores. Los procesos pueden ser tanto servidores como clientes. Por ejemplo, un programa de aplicación normal es un cliente que llama al servidor correspondiente para acceder a un archivo o realizar una operación de

entrada/salida sobre un dispositivo concreto. A su vez, un proceso cliente puede actuar como servidor para otro.

Este paradigma ofrece gran flexibilidad en cuanto a los servicios posibles en el sistema final, ya que el núcleo provee solamente funciones muy básicas de memoria, entrada/salida, archivos y procesos, dejando a los servidores proveer la mayoría que el usuario final o programador puede usar. Estos servidores deben tener mecanismos de seguridad y protección que, a su vez, serán filtrados por el núcleo que controla el hardware.

Gráfica 17. Cliente-servidor



2.2.2 Sistemas operativos por los modos de explotación¹³

Los modos de explotación se corresponden con las distintas maneras en que puede funcionar un sistema operativo. Dentro de ellas, se encuentran las indicadas en los apartados siguientes.

a. Procesamiento por lotes

Los sistemas operativos por lotes, procesan una gran cantidad de trabajos con poca o ninguna interacción entre los usuarios y los programas en ejecución. Se reúnen todos los trabajos comunes para realizarlos al mismo tiempo, evitando la espera de dos o más trabajos como sucede en el procesamiento en serie. Estos sistemas son de los más tradicionales y antiguos, y fueron introducidos alrededor de 1956 para aumentar la capacidad de procesamiento de los programas.

Cuando estos sistemas son bien planeados, pueden tener un tiempo de ejecución muy alto, porque el procesador es mejor utilizado y los sistemas operativos pueden ser simples, debido a la secuenciabilidad de la ejecución de los trabajos.

Algunos ejemplos de Sistemas Operativos por lotes exitosos son el SCOPE, del DC6600, el cual está orientado a procesamiento científico pesado, y el EXEC II para el UNIVAC 1107, orientado a procesamiento académico.

¹³ RAYA, L. ALVAREZ, R. RODRIGO, V. (2.005). *Sistemas Operativos en entornos Monousuario y Multiusuario*. México, Alfaomega, Ra-Ma. p. 21.

Algunas otras características con que cuentan los sistemas operativos por lotes son:

- Requiere que el programa, datos y órdenes al sistema sean remitidos todos juntos en forma de lote.
- Permiten poca o ninguna interacción usuario/programa en ejecución.
- Mayor potencial de utilización de recursos que procesamiento serial simple en sistemas multiusuarios.
- No conveniente para desarrollo de programas por bajo tiempo de retorno y depuración fuera de línea.
- Conveniente para programas de largos tiempos de ejecución (ej, análisis estadísticos, nóminas de personal, etc.).
- Se encuentra en muchos computadores personales combinados con procesamiento serial.
- Planificación del procesador sencilla, típicamente procesados en orden de llegada.
- Planificación de memoria sencilla, generalmente se divide en dos: parte residente del S.O. y programas transitorios.
- No requieren gestión crítica de dispositivos en el tiempo.
- Suelen proporcionar gestión sencilla de manejo de archivos: se requiere poca protección y ningún control de concurrencia para el acceso.

b. Multiprogramación

En este modo de explotación, el sistema operativo se encarga de distribuir la carga computacional entre los procesadores existentes (monoprocesador o multiprocesador), con el fin de incrementar el poder de procesamiento de la máquina.

Dentro de los sistemas operativos multiprogramados cabe diferenciar:

- **Tiempo compartido.** Permiten la simulación de que el sistema y sus recursos son todos para cada usuario. El usuario hace una petición a la computadora, esta la procesa tan pronto como le es posible, y la respuesta aparecerá en la terminal del usuario.

Los principales recursos del sistema, el procesador, la memoria, dispositivos de E/S, son continuamente utilizados entre los diversos usuarios, dando a cada usuario la ilusión de que tiene el sistema dedicado para sí mismo. Esto trae como consecuencia una gran carga de trabajo al sistema operativo, principalmente en la administración de memoria principal y secundaria.

Utilizan las distintas técnicas de planificación de CPU para que se atiendan todos los procesos en espera de ser ejecutados. Este proceso ocurre tan rápidamente que el usuario no lo percibe.

Entre este tipo de sistemas operativos se encuentran: UNIX, LINUX Windows 95, Windows 98, Windows Millenium, Windows XP, Windows NT, Windows 2000, Windows 2003, MACOS y OS/2. Otros menos comunes son Multics, OS/360 y DEC-10.

Las características de los sistemas operativos de tiempo compartido pueden ser:

- Son populares representantes de sistemas multiprogramados multiusuario, ejemplo: sistemas de diseño asistido por computador, procesamiento de texto, etc.
- Dan la ilusión de que cada usuario tiene una máquina para sí.
- La mayoría utilizan algoritmo de reparto circular.
- Los programas se ejecutan con prioridad rotatoria que se incrementa con la espera y disminuye después de concedido el servicio.
- Evitan la monopolización del sistema asignando tiempos de procesador (time slot).
- La gestión de memoria proporciona protección a programas residentes.
- La gestión de archivo debe proporcionar protección y control de acceso debido a que pueden existir múltiples usuarios accediendo un mismo archivo.

- **Tiempo real.** Un sistema en tiempo real es aquel en el cual los resultados son correctos no solo si la computación es correcta, sino que también ha de serlo el tiempo en el cual se producen los resultados.

Los sistemas operativos de tiempo real son aquellos en los cuales no tiene importancia el usuario, sino los procesos. Por lo general, están subutilizados sus recursos con la finalidad de prestar atención a los procesos en el momento que lo requieran. Se utilizan en entornos donde son procesados un gran número de sucesos o eventos.

Muchos sistemas operativos de tiempo real son construidos para aplicaciones muy específicas como control de tráfico aéreo, bolsas de valores, control de refinerías, control de laminadores. También en el ramo automovilístico y de la electrónica de consumo, las aplicaciones de tiempo real están creciendo muy rápidamente.

Otros campos de aplicación de los sistemas operativos de tiempo real son los siguientes:

- Control de trenes.
- Telecomunicaciones.
- Sistemas de fabricación integrada.
- Producción y distribución de energía eléctrica.
- Control de edificios.
- Sistemas multimedia.

Algunos ejemplos de sistemas operativos de tiempo real son: VxWorks, Solaris, Lyns OS y Spectra.

Los sistemas operativos de tiempo real, cuentan con las siguientes características:

- Se dan en entornos en donde deben ser aceptados y procesados gran cantidad de sucesos, la mayoría externos al sistema computacional, en breve tiempo o dentro de ciertos plazos.
- Se utilizan en control industrial, conmutación telefónica, control de vuelo, simulaciones en tiempo real, aplicaciones militares, etc.
- Su objetivo es proporcionar rápidos tiempos de respuesta.
- Procesa ráfagas de miles de interrupciones por segundo sin perder un solo suceso.
- El proceso se activa tras ocurrencia de suceso, mediante interrupción.
- El proceso de mayor prioridad expropia recursos. Por tanto generalmente se utiliza planificación expropiativa basada en prioridades.
- La gestión de memoria es menos exigente que en tiempo compartido, usualmente los procesos son residentes permanentes en memoria.
- La población de procesos es estática en gran medida.
- Existe poco movimiento de programas entre almacenamiento secundario y memoria.
- La gestión de archivos se orienta más a velocidad de acceso que a utilización eficiente del recurso.

Para ejecutar un conjunto de tareas concurrentes con un único procesador hace falta multiplexar el uso del mismo entre todas las tareas activas en un momento dado. Utilizar algoritmos de planificación equitativos (como Round Robin) no permite garantizar el tiempo de respuesta de las tareas. Para solucionar este problema se ha de utilizar la planificación basada en prioridades. (Estos algoritmos se detallan más adelante)

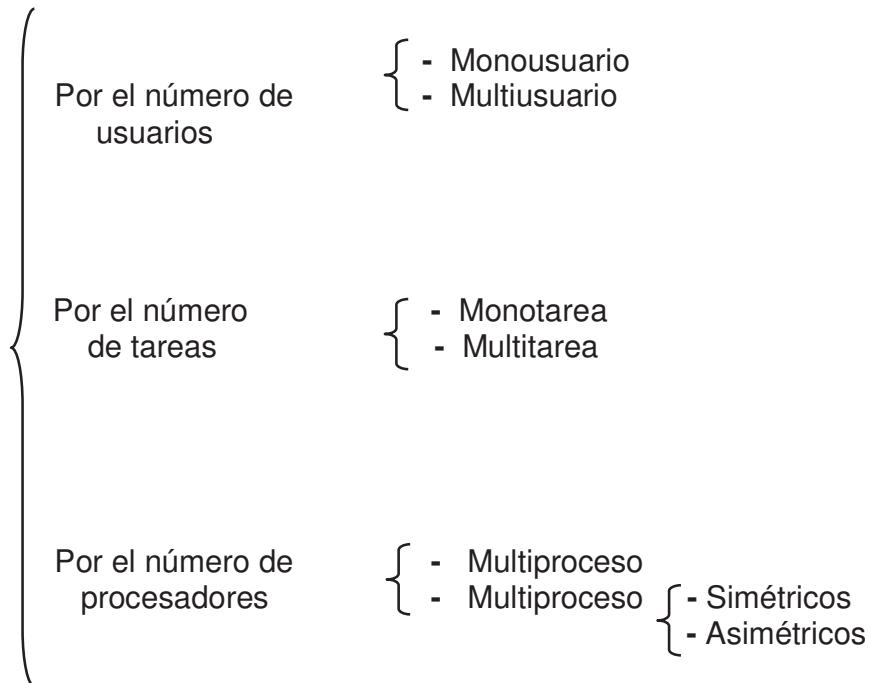
Los sistemas operativos en tiempo real son sistemas muy complejos que suelen diseñarse a medida para ciertas aplicaciones, después de mucho tiempo de estudio de todas las opciones y problemas que pudieran surgir.

- **Híbrido.** Estos sistemas operativos intentan ser una mezcla de los dos anteriores, buscando combinar las ventajas de los sistemas en tiempo compartido y en tiempo real.

2.2.3 Sistemas operativos por los servicios ofrecidos¹⁴

En esta clasificación se tiene en cuenta la visión del usuario final y puede ser la siguiente:

¹⁴ Ibid. p. 22.



a. Monousuarios

Los sistemas operativos **monousuario** son aquellos que únicamente soportan 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. Las computadoras personales típicamente se han clasificado en este renglón.

b. Multiusuarios

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.

c. Monotareas

Los sistemas operativos **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 sólo una tarea a la vez.

d. Multitareas

Un sistema operativo **multitarea** es aquel que permite al usuario estar realizando varios trabajos 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. Es común encontrar en ellos interfaces graficas orientadas al uso de menús y al ratón, lo que permite un rápido intercambio entre las tareas para el usuario, mejorando para su productividad.

e. Monoproceso o uniproceso

Los sistemas **monoproceso** son los que únicamente permiten realizar un proceso a la vez. Sin embargo, permiten simular la multitarea haciendo que el sistema realice una tarea rotatoria con intercambio muy rápido. Ejemplos típicos de este tipo de sistemas son el DOS y MacOS.

f. Multiproceso

Los sistemas operativos **multiproceso** son los que permiten realizar varios procesos simultáneamente y, por tanto, son capaces de ejecutar varias tareas al mismo tiempo.

Dentro de los sistemas multiproceso, se encuentran los sistemas **simétricos**, que son los que distribuyen la carga de procesamiento por igual entre todos los procesadores existentes. Sin embargo, los sistemas multiproceso **asimétricos**, como Windows NT, 2000 y 2003 Server, asignan una tarea por procesador existente, según su prioridad, y el resto de tareas (de baja prioridad) se ejecutan en un único procesador. Por ejemplo, un sistema biprocesador asimétrico ejecutaría una sola tarea en un procesador y el resto en el otro.

Cuando se trabaja de manera asimétrica, 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.

Cuando se trabaja de manera simétrica, los procesos o partes de ellos (threads) son enviados indistintamente a cualquiera de los procesadores disponibles, teniendo, teóricamente, una mejor distribución y equilibrio en la carga de trabajo bajo este esquema. Se dice que un thread es la parte activa en memoria y corriendo de un proceso, lo cual puede consistir de un área de memoria, un conjunto de registros con valores específicos, la pila y otros valores de contexto.

Un aspecto importante a considerar en estos sistemas es la forma de crear aplicaciones para aprovechar los procesadores. Existen aplicaciones que fueron hechas para correr en sistemas monoproceso que no toman ninguna ventaja a menos que el sistema operativo o el compilador detecte secciones de código paralelizable, los cuales son ejecutados al mismo tiempo en procesadores diferentes. Por otro lado, el programador puede modificar sus algoritmos y aprovechar por sí mismo esta facilidad; pero esta última opción es costosa en horas hombre y muy tediosa, obligando al programador a ocupar tanto o más tiempo a la paralelización que a elaborar el algoritmo inicial.

2.2.4 Por la forma de ofrecer los servicios¹⁵

En esta clasificación se encuentran:

a. Sistemas centralizados

Hasta que los computadores personales no tuvieron un precio accesible y suficiente potencia, la mayoría de los sistemas (UNIX) utilizaban el modelo de proceso centralizado. Con este tipo de modelo los computadores mainframe se encargaban de todo el procesamiento y los usuarios manejaban únicamente terminales brutas(es decir, no disponían de memoria, ni procesador).

Actualmente se siguen utilizando los sistemas centralizados (como los Terminal Services de Microsoft) pero las terminales dejan de ser brutas y pueden realizar otras muchas tareas por sí mismas.

Los principales sistemas operativos centralizados en el mercado son: z/OS, OS/390, Linux, TPF, VSE y ESA.

b. Sistemas de red

Estos sistemas operativos son aquellos que mantienen a dos o más computadores unidas a través de algún medio de comunicación (físico o no), con el objetivo primordial de poder compartir los diferentes recursos y la información del sistema.

En este entorno, cada computador mantiene su propio sistema operativo y su propio sistema de archivos local.

El primer sistema operativo de red estaba enfocado a equipos con un procesador *Motorola 68000*, pasando posteriormente a procesadores Intel.

¹⁵ Ibid. p. 23.

Los sistemas operativos de red usados más ampliamente son: Novell NetWare, Personal NetWare, LAN Manager, Windows NT Server, Windows 2000 Server, UNIX, LINUX, LANTastic, etc.

c. Sistemas distribuidos

Los sistemas operativos distribuidos son sistemas quasi-independientes que permiten distribuir los trabajos, tareas o procesos entre un conjunto de procesadores. Puede ocurrir que este conjunto de procesadores se encuentre en el mismo equipo o en equipos distintos (siendo, en este último caso, transparente para el usuario).

Existen dos esquemas básicos:

- Un sistema **fuertemente acoplado** es aquel que comparte la memoria y un reloj global, cuyos tiempos de acceso son similares para todos los procesadores.
- Un sistema **débilmente acoplado** es aquel en el que los procesadores no comparten ni memoria ni reloj, ya que cada uno de ellos cuenta con memoria local.

Las principales ventajas de los sistemas distribuidos (no solamente los sistemas operativos) con respecto a los sistemas centralizados se describen enseguida:

- **Economía:** El cociente precio/desempeño de la suma del poder de los procesadores separados contra el poder de uno solo centralizado es mejor cuando están distribuidos.
- **Velocidad:** Relacionado con el punto anterior, la velocidad sumada es muy superior.
- **Confiabilidad:** Si una sola máquina falla, el sistema total sigue funcionando.
- **Crecimiento:** El poder total del sistema puede irse incrementando al añadir pequeños sistemas, lo cual es mucho más difícil en un sistema centralizado y caro.
- **Distribución:** Algunas aplicaciones requieren de por sí una distribución física.

Por otro lado, los sistemas distribuidos también exhiben algunas ventajas sobre sistemas aislados. Estas ventajas son:

- **Compartir datos:** Un sistema distribuido permite compartir datos más fácilmente que los sistemas aislados, que tendrían que duplicarlos en cada nodo para lograrlo.

- **Compartir dispositivos:** Un sistema distribuído permite accesar dispositivos desde cualquier nodo en forma transparente, lo cual es imposible con los sistemas aislados. El sistema distribuído logra un efecto sinergético.
- **Comunicaciones:** La comunicación persona a persona es factible en los sistemas distribuídos, en los sistemas aislados no.
- **Flexibilidad:** La distribución de las cargas de trabajo es factible en el sistema distribuído, se puede incrementar el poder de cómputo.

Así como los sistemas distribuídos exhiben grandes ventajas, también se pueden identificar algunas desventajas, algunas de ellas tan serias que han frenado la producción comercial de sistemas operativos en la actualidad.

El problema más importante en la creación de sistemas distribuídos es el software: los problemas de compartición de datos y recursos es tan complejo que los mecanismos de solución generan mucha sobrecarga al sistema haciéndolo ineficiente. El chequear, por ejemplo, quiénes tienen acceso a algunos recursos y quiénes no, el aplicar los mecanismos de protección y registro de permisos consume demasiados recursos. En general, las soluciones presentes para estos problemas están aún en pañales.

Otros problemas de los sistemas operativos distribuídos surgen debido a la concurrencia y al paralelismo. Tradicionalmente las aplicaciones son creadas para computadoras que ejecutan secuencialmente, de manera que el identificar secciones de código "paralelizable" es un trabajo árduo, pero necesario para dividir un proceso grande en sub-procesos y enviarlos a diferentes unidades de procesamiento para lograr la distribución. Con la concurrencia se deben implantar mecanismos para evitar las condiciones de competencia, las postergaciones indefinidas, el ocupar un recurso y estar esperando otro, las condiciones de espera circulares y, finalmente, los "abrazos mortales" (deadlocks). Estos problemas de por sí se presentan en los sistemas operativos multiusuarios o multitareas, y su tratamiento en los sistemas distribuídos es aún más complejo, y por lo tanto, necesitará de algoritmos más complejos con la inherente sobrecarga esperada.

Los sistemas operativos distribuidos más extendidos son los siguientes: Sprite, Solaris-MC, Mach, Chorus, Spring, Amoeba, Taos, etc.

d. Sistemas operativos paralelos

En estos tipos de sistemas operativos se pretende que cuando existan dos o más procesos que compitan por algún recurso se puedan realizar o ejecutar al mismo tiempo.

En UNIX existe también la posibilidad de ejecutar programas sin tener que atenderlos en forma interactiva, simulando paralelismo (es decir, atender de manera concurrente varios procesos de un mismo usuario). Así, en lugar de esperar a que el proceso termine de ejecutarse (como lo haría normalmente), regresa a atender al usuario inmediatamente después de haber creado el proceso. Ejemplos de este tipo de sistemas operativos están: Alpha, PVM, la serie AIX, que es utilizado en los sistemas RS/6000 de IBM.

2.3 Estructura de los sistemas operativos

Si bien no todos los sistemas operativos tienen la misma estructura, la mayoría de los sistemas operativos modernos poseen una misma estructura.

El Kernel consiste en la parte principal del código del sistema operativo, el cual se encarga de controlar y administrar los servicios y peticiones de recursos y de hardware con respecto a uno o varios procesos.

En los diseños en que el núcleo está distribuido en varios niveles de jerarquía, elegir qué función colocar en cada nivel requiere un análisis cuidadoso. En tales diseños, con frecuencia sólo se permite hacer llamadas a funciones situadas jerárquicamente por debajo de quien hace la llamada; es decir, cada nivel sólo puede llamar a las funciones que están colocadas en el nivel inmediato inferior. (Recordar gráfica 15. Estructura por capas o jerárquica)

De esta forma una estructura general de un sistema operativo sería:

Arriba del núcleo, en jerarquía, se encuentran los diferentes procesos del sistema operativo que trabajan en apoyo de los procesos de usuario, que se encargan en la práctica de supervisar las operaciones de entrada/salida de los dispositivos del sistema para beneficio de los diversos usuarios. Esta estructura sería:

a. Administrador de procesos

Un programa no hace nada a menos que sus instrucciones sean ejecutadas por la CPU. Un proceso necesita ciertos recursos, tiempo de CPU, memoria, archivos y dispositivos de E/S, para completar sus tareas. Estos recursos son reservados cuando se crea el proceso o bien se otorgan en tiempo de ejecución.

El sistema operativo es responsable de:

- La creación y eliminación de procesos de sistema y de usuarios.
- Detener y continuar ejecutando un proceso.
- Proveer mecanismos para sincronizar procesos.
- Proveer mecanismos para comunicar procesos.
- Proveer mecanismos para proteger procesos.

b. Administrador de memoria

El procesador central lee y escribe datos directamente en memoria. La memoria principal es generalmente el único dispositivo de almacenamiento que la CPU puede acceder directamente. Por ejemplo para que la CPU procese datos del disco, primero se deben cargar éstos en la memoria.

El sistema operativo es responsable de:

- Mantener historial de las partes de memoria que pueden ser accedidas concurrentemente y que procesos pueden hacerlo.
- Decidir qué procesos se cargarán en memoria cuando se encuentre lugar disponible en ésta.
- Asignar y quitar espacio de memoria según las necesidades.

c. Administrador de almacenamiento secundario

Como la memoria principal es muy chica como para almacenar todos los datos y programas necesarios, la computadora posee un almacenamiento secundario para volcar los datos de memoria no utilizados. Las computadoras modernas utilizan el disco para este fin. La mayoría de los programas se almacenan en disco hasta que son cargados en memoria.

El sistema operativo es responsable de:

- Administrar el espacio libre.
- Asignar espacio de almacenamiento.
- Organizar el disco.

d. Administrador de sistemas de E/S

Uno de los propósitos del sistema operativo es ocultar las peculiaridades de los dispositivos de hardware al usuario.

Los sistemas de E/S consisten de:

- Un sistema de buffer intermedio.
- Una interfaz general.
- Manejadores de dispositivos de hardware específicos.

e. Administrador de archivos

El administrador de archivos es uno de los componentes más visibles de un sistema operativo. Las computadoras pueden almacenar información en diferentes tipos de medios físicos. Cintas magnéticas, discos magnéticos y discos ópticos, son los más comunes. Cada uno de estos medios tiene sus propias características

y organización física. Cada medio se controla por un dispositivo. Las propiedades incluyen velocidad, capacidad, velocidad de transferencia de datos y método de acceso (Secuencial o Random).

Por conveniencia el sistema operativo provee una vista lógica uniforme de la información, independientemente de las características de cada dispositivo, utiliza la unidad **archivo**.

Un archivo es un grupo de información relacionada definida por su creador (programas o datos).

El sistema operativo es responsable de:

- Creación y eliminación de archivos.
- Creación y eliminación de directorios.
- Soporte de primitivas (instrucciones) para manipular archivos y directorios.
- Mapeo de archivos dentro de almacenamiento secundarios.
- Resguardar archivos en medios de almacenamiento estables.

f. Sistema de protección

Si un sistema tiene múltiples usuarios y permite múltiples usuarios concurrentes, los procesos deben estar protegidos de otras actividades. Para tal propósito se provee de mecanismos que aseguran que los archivos, segmentos de memoria, CPU y otros recursos pueden ser operados sólo por aquellos procesos que tienen permiso otorgado por el sistema operativo.

Por **protección** se entiende a los mecanismos para controlar el acceso de programas, procesos y usuario a los recursos definidos por el sistema.

g. Networking

Un sistema distribuido es una colección de procesos que no comparten memoria o recursos. Cada procesador tiene su propia memoria local, y los procesadores se comunican con otros a través de varias líneas de comunicación como ser buses de alta velocidad o líneas telefónicas.

Los procesadores en el sistema se conectan a través de redes de comunicación, las cuales se pueden configurar de muchas maneras. La red puede estar completa o parcialmente conectada.

En un sistema distribuido los recursos se comparten entre varias estaciones, los sistemas operativos de red se encargan de administrar el acceso a estos recursos.

h. Sistema intérprete de comandos

Uno de las funciones más importantes de un sistema operativo es el intérprete de comandos, que es la interfaz entre el usuario y el sistema operativo. Algunos sistemas operativos incluyen el intérprete en el kernel. Otros como el DOS o UNIX, poseen un programa especial para cumplir esta función que se ejecuta cuando se inicia el sistema.

Los sistemas operativos se diferencian en el área de interpretación de comandos, según el grado de facilidad que presenten a los usuarios. Por ejemplo en Windows para copiar un archivo de una unidad a otra el usuario puede seleccionar con el mouse el archivo que desea copiar y arrastrarlo hasta su nuevo destino; mientras que en DOS, debe ingresar una sentencia desde una pantalla de caracteres

2.4 Núcleos de sistemas operativos

El núcleo (*Kernel*) de un sistema operativo es un conjunto de rutinas cuya misión es la de gestionar el procesador, la memoria, la entrada/salida y el resto de procesos disponibles en la instalación. Toda esta gestión la realiza para atender al funcionamiento y peticiones de los trabajos que se ejecutan en el sistema.

Todas las operaciones en las que participan procesos son controladas por la parte del sistema operativo denominada núcleo (nucleus, core o kernel, en inglés). El núcleo normalmente representa sólo una pequeña parte de lo que por lo general se piensa que es todo el sistema operativo, pero es tal vez el código que más se utiliza. Por esta razón, el núcleo reside por lo regular en la memoria principal, mientras que otras partes del sistema operativo son cargadas en la memoria principal sólo cuando se necesitan.

Los núcleos se diseñan para realizar el mínimo posible de procesamiento en cada interrupción y dejar que el resto lo realice el proceso apropiado del sistema, que puede operar mientras el núcleo se habilita para atender otras interrupciones.

2.4.1 Funciones del núcleo

El núcleo de un sistema operativo normalmente contiene el código necesario para realizar las siguientes funciones:

- Manejo de interrupciones.
- Creación y destrucción de procesos.
- Cambio de estado de los procesos.
- Despacho.
- Suspensión y reanudación de procesos.
- Sincronización de procesos.
- Comunicación entre procesos.
- Manipulación de los bloques de control de procesos.

- Apoyo para las actividades de entrada/salida.
- Apoyo para asignación y liberación de memoria.
- Apoyo para el sistema de archivos.
- Apoyo para el mecanismo de llamada y retorno de un procedimiento.
- Apoyo para ciertas funciones de contabilidad del sistema.

2.4.2 Categorías de los núcleos

Los núcleos (kernels) de los sistemas operativos se pueden ubicar en dos categorías:

- Monolíticos o,
- Micronúcleos (microkernels).

El primer tipo de núcleo es el usado tradicionalmente, mientras que los micronúcleos forman parte de las tendencias modernas en el diseño de sistemas operativos.

Para comprender mejor qué diferencias existen entre ambas categorías, vamos a revisar algunos conceptos:

a. Trabajos, Procesos y Thread

Estos tres conceptos van definiendo el grado de granularidad en que el sistema operativo trata a las masas de operaciones que se tienen que realizar. Un trabajo se conceptualiza como un conjunto de uno o más procesos. Por ejemplo, si se tiene que hacer el trabajo de correr el inventario, tal vez se subdivida ese trabajo en varios procesos: obtener la lista de artículos, número en existencia, artículos vendidos, artículos extraviados, etc. Un proceso se define como la imagen de un programa en ejecución, es decir, en memoria y usando el CPU. A este nivel de granularidad, un proceso tiene un espacio de direcciones de memoria, una pila, sus registros y su “program counter”. Un thread es un trozo o sección de un proceso que tiene sus propios registros, pila y “program counter” y puede compartir la memoria con todos aquellos threads que forman parte del mismo proceso.

b. Objetos

Un objeto es una entidad que contiene dos partes principales: una colección de atributos y un conjunto de métodos (también llamados servicios). Generalmente los atributos del objeto no pueden ser cambiados por el usuario, sino solamente a través de los métodos. Los métodos sí son accesibles al usuario y de hecho es lo único que él observa: los métodos conforman lo que se llama la “interfaz” del objeto. Por ejemplo, para el objeto “archivo” los métodos son abrir, cerrar, escribir, borrar, etc. El cómo se abre, se cierra, se borra, etc; está escondido para el

usuario, es decir, los atributos y el código están “encapsulados”. La única forma de activar un método es a través del envío de mensajes entre los objetos, o hacia un objeto.

c. Cliente - Servidor

Un cliente es un proceso que necesita de algún valor o de alguna operación externa para poder trabajar. A la entidad que provee ese valor o realiza esa operación se le llama servidor. Por ejemplo, un servidor de archivos debe correr en el núcleo (kernel) o por medio de un proceso “guardián” al servidor de archivos que escucha peticiones de apertura, lectura, escritura, etc; sobre los archivos. Un cliente es otro proceso guardián que escucha esas peticiones en las máquinas clientes y se comunica con el proceso servidor a través de la red, dando la apariencia de que se tienen los archivos en forma local en la máquina cliente.

d. Núcleo Monolítico

Los núcleos monolíticos generalmente están divididos en dos partes estructuradas: el núcleo dependiente del hardware y el núcleo independiente del hardware. El núcleo dependiente se encarga de manejar las interrupciones del hardware, hacer el manejo de bajo nivel de memoria y discos y trabajar con los manejadores de dispositivos de bajo nivel, principalmente. El núcleo independiente del hardware se encarga de ofrecer las llamadas al sistema, manejar los sistemas de archivos y la planificación de procesos. Para el usuario esta división generalmente pasa desapercibida. Para un mismo sistema operativo corriendo en diferentes plataformas, el núcleo independiente es exactamente el mismo, mientras que el dependiente debe re-escibirse.

e. Microkernel

Un núcleo con “arquitectura” micronúcleo es aquel que contiene únicamente el manejo de procesos y threads, el de manejo bajo de memoria, da soporte a las comunicaciones y maneja las interrupciones y operaciones de bajo nivel de entrada-salida. En los sistemas operativos que cuentan con este tipo de núcleo se usan procesos “servidores” que se encargan de ofrecer el resto de servicios (por ejemplo el de sistema de archivos) y que utilizan al núcleo a través del soporte de comunicaciones.

Este diseño permite que los servidores no estén atados a un fabricante en especial, incluso el usuario puede escoger o programar sus propios servidores. La mayoría de los sistemas operativos que usan este esquema manejan los recursos de la computadora como si fueran objetos: los servidores ofrecen una serie de “llamadas” o “métodos” utilizables con un comportamiento coherente y estructurado.

Otra de las características importantes de los micronúcleos es el manejo de threads. Cuando un proceso está formado de un solo thread, éste es un proceso normal como en cualquier sistema operativo.

Los usos más comunes de los micronúcleos son en los sistemas operativos que intentan ser distribuidos, y en aquellos que sirven como base para instalar sobre ellos otros sistemas operativos. Por ejemplo, el sistema operativo AMOEBA intenta ser distribuido y el sistema operativo MACH sirve como base para instalar sobre él DOS, UNIX, etc.

Actividad de refuerzo:

Claro el tema abordado? Bien.

Entonces ahora saque las dos listas, por cada sistema operativo, que realizó en la actividad inicial, revise nuevamente las características y desventajas que había descrito de los sistemas operativos Windows y Linux (versiones seleccionadas). Si cree que debe cambiar algo en cuaquiera de las listas, o al contrario agregar algún otro ítem lo puede hacer.

Socialice su ejercicio con el tutor. Resalte los puntos en los que no tuvo aciertos.

CAPÍTULO 3. INTERBLOQUEO

Activación cognitiva:

Reúnanse en grupos de trabajo, no importa el número de integrantes, sólo importa que el grupo tenga mínimo 2 estudiantes. Y analicen qué es el interbloqueo y en qué actividades de la vida cotidiana se pueden presentar situaciones de Interbloqueo. Comentar en grupo, justificar sus respuestas.

3.1 Recursos¹⁶

Un sistema se compone de un número finito de recursos que son distribuidos entre un número de procesos que compiten por ellos.

Existen recursos:

- **Físicos:** ciclos de CPU, espacio en memoria, dispositivos de E/S (impresoras, unidades de cinta, etc.).
- **Lógicos:** ficheros, tablas del sistema, semáforos.

Los recursos son clasificados en diferentes tipos, cada uno de los cuales se compone de algún número de instancias iguales.

Si un sistema tiene dos CPU's, entonces el tipo CPU tiene dos instancias.

Similarmente, el tipo IMPRESORAS puede tener cinco instancias. Si un proceso pide una instancia de un tipo de recurso, la asignación de cualquier instancia de ese tipo atenderá la petición.

Si este no es el caso, entonces las instancias no son idénticas y las clases de tipos de recursos no están bien definidas. Un proceso debe solicitar un recurso antes de usarlo y liberarlo después de usarlo. Un proceso puede solicitar tantos recursos como sean necesarios para llevar a cabo la tarea para la cual ha sido diseñado. Obviamente el número de recursos solicitados no debe exceder el número de recursos disponibles en el sistema. En otras palabras, un proceso no debe pedir tres impresoras si en el sistema solo existen dos.

Los bloqueos pueden ocurrir cuando los procesos obtienen el acceso exclusivo a los dispositivos, archivos, etc. Para analizar los bloqueos de manera más general, nos referimos a los objetos conocidos como recursos.

¹⁶ Sistemas Operativos II. Instituto Tecnológico de Veracruz. <http://www.itver.edu.mx/so2/>
Unidad II. Tema 2.2.

Como se mencionó, un recurso puede ser cualquier dispositivo de hardware (unidad de cinta) o una parte de información (por ejemplo un registro cerrado en una base de datos). Una computadora normalmente tendrá varios recursos que pueden ser otorgados. Algunos podrán tener varias referencias idénticas, como en el caso de las unidades de cinta. Si se tienen varias copias disponibles de un recurso, cualquiera de ellas se puede utilizar para satisfacer cualquier solicitud de recurso.

En resumen, un recurso es cualquier elemento que pueda ser utilizado por un único proceso en un momento dado.

Los recursos son de dos tipos:

- Apropiativos
- No apropiativos

3.1.1 Recursos apropiativos

Gráfica 18. Recursos apropiativos. Memoria.



Los recursos apropiativos son aquellos que se pueden tomar del proceso que le posee sin efectos dañinos. La memoria es un ejemplo de recursos apropiativos.

Por ejemplo consideremos un sistema con 512 Kb de memoria de usuario, una impresora, y dos procesos de 512 Kb que se desean imprimir cada uno.

- El proceso A solicita y obtiene la impresora, para entonces comienza a calcular los valores que va a imprimir. Antes de terminar el cálculo excede su quantum de tiempo y se intercambia.
- El proceso B se empieza a ejecutar e intenta sin éxito adquirir la impresora.
- En potencia tendríamos una situación de bloqueo, puesto que A tiene la impresora y B la memoria y ninguno puede continuar sin el recurso que posee el otro.
- Por fortuna es posible apropiarse de la memoria de B y sacarlo de ella e intercambiárselo con A.

- A puede ejecutarse entonces, imprimir y liberar después la impresora. No ocurre un bloqueo.

3.1.2 Recursos no apropiativos

Gráfica 19. Recursos no apropiativos. Impresora.



Los recursos no apropiativos son aquellos que no se pueden tomar de su poseedor activo sin provocar un fallo de cálculo. Por ejemplo, si un proceso comienza a imprimir una salida, se toma la impresora y se le da otro proceso, el resultado será una salida incomprendible. Las impresoras no son apropiables.

En general los bloqueos se relacionan con los recursos no apropiables.

Bajo un modo normal de operación un proceso puede utilizar un recurso sólo en la siguiente secuencia:

1. **Petición.** Si la petición no puede ser satisfecha inmediatamente (por ejemplo, el recurso está siendo utilizado por otro proceso), entonces el proceso solicitante debe esperar hasta que pueda adquirir el recurso.
2. **Uso.** El proceso puede utilizar un recurso (por ejemplo, si el recurso es una impresora en línea, el proceso puede imprimir en la impresora).
3. **Liberación.** El proceso libera el recurso. Si el recurso no está disponible cuando es requerido, el proceso solicitante se ve forzado a esperar. En algunos sistemas operativos, el proceso se bloquea automáticamente cuando falla la solicitud el recurso y se desbloquea cuando está disponible. En otros sistemas, la requisición falla en un código de error y corresponde al recurso solicitante esperar un poco y volverlo a intentar.

La petición y liberación del recurso son peticiones al sistema. Ejemplos de llamadas al sistema son: Petición/Liberación de dispositivos, Abrir/Cerrar archivos y Asignar/Desasignar memoria.

El uso de recursos puede también hacerse sólo a través de llamadas al sistema. Por lo tanto, para cada uso, el sistema operativo chequea para asegurarse de que el proceso usuario ha pedido y se le han asignado los recursos. Una tabla del sistema registra cuando un recurso está libre o asignado, y si está asignado, a qué proceso. Si un proceso pide un recurso que está asignado en ese momento a otro

proceso, éste puede ser agregado a una cola de procesos en espera de ese recurso.

3.2 Análisis¹⁷

Gráfica 20. Recursos



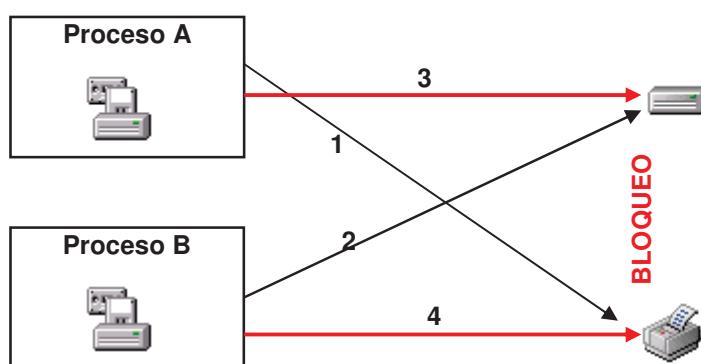
Los sistemas de cómputo tienen muchos recursos que sólo pueden ser utilizados por un proceso a la vez. Los ejemplos más comunes son las impresoras, las unidades de cinta y los espacios de las tablas de archivos. Si dos procesos utilizaran al mismo tiempo el mismo espacio en la tabla de archivos, se tendría un sistema de archivo corrupto. Por ello, todos los sistemas operativos tienen la capacidad de otorgar a un proceso (en forma temporal) el acceso exclusivo a ciertos recursos.

En muchas aplicaciones, un proceso necesita el acceso exclusivo no solo a un recurso, sino a varios recursos. Un proceso que copie un archivo mayor que el tamaño de un disco de una cinta magnética a una impresora necesita el acceso exclusivo a la unidad de cinta y a la impresora al mismo tiempo. En un sistema monousuario, éste puede tener acceso a todos los recursos que necesite y realizar su trabajo.

Sin embargo, en un sistema multiusuario, pueden surgir serios problemas.

3.2.1 Bloqueos

Gráfica 21. Ejemplo de bloqueo



¹⁷ Ibid. Unidad II. Temas 2.1 y 2.3.

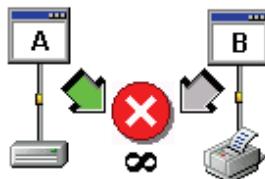
Veamos:

- Supongamos que dos procesos desean imprimir cada uno un enorme archivo en cinta. El proceso A solicita permiso para utilizar la impresora, el cual se le concede.
- Entonces cuando el proceso B solicita permiso para utilizar la unidad de cinta y se le otorga.
- El proceso A solicita entonces la unidad de cinta, pero la solicitud es denegada hasta que B la libere.
- Por desgracia, en este momento, en vez de liberar la unidad de cinta, el proceso B solicita la impresora.
- Los procesos se bloquean en ese momento y permanecen así por siempre.

A esta situación se le llama **BLOQUEO**.

3.2.2 Abrazo Mortal (Deadlock)

Gráfica 22. Ejemplo de deadlock



En ambientes de multiprogramación, varios procesos pueden competir por un número finito de recursos. Un proceso solicita recursos, y si los recursos no están disponibles en ese momento, el proceso entra en un estado de espera. Puede suceder que los procesos en espera nunca cambien de estado, debido a que los recursos que han solicitado están siendo detenidos por otros procesos en espera. Por ejemplo, esta situación ocurre en un sistema con cuatro unidades de disco y dos procesos. Si cada proceso tiene asignadas dos unidades de disco pero necesita tres, entonces cada proceso entrará a un estado de espera, en el cual estará hasta que el otro proceso libere las unidades de cinta que tiene asignadas.

Esta situación es llamada **Abrazo Mortal** (Deadlock). Para prevenir un abrazo mortal o recuperarse de alguno que ocurra, el sistema puede tomar alguna acción relativamente extrema, tal como el derecho de tomar los recursos (preemption of resources) de uno o más de los procesos que se encuentren en el abrazo mortal.

A lo largo del capítulo se describen algunos de los métodos que pueden utilizarse en un sistema operativo para manejar el problema de abrazo mortal.

El problema del abrazo mortal (Deadlock)

El problema de los abrazos mortales no es único al ambiente de los sistemas operativos. Generalizando nuestra interpretación de recursos y procesos, podemos ver que un problema de abrazo mortal puede ser parte de nuestro ambiente de vida diaria.

Por ejemplo, considere el problema de cruzar un río que tiene un número de piedras de apoyo. A lo más, un pie puede estar pisando una piedra a la vez. Para cruzar el río, cada persona debe de utilizar cada una de las piedras de apoyo.

Podemos suponer que cada una de las personas que cruzan el río es un proceso y que cada piedra de apoyo es un recurso. Un abrazo mortal ocurre cuando dos personas intentan cruzar el río desde orillas opuestas y se encuentran en el medio.

El pisar una piedra puede ser vista como el adquirir un recurso, mientras que el quitar el pie de la piedra corresponde a liberar el recurso. Un abrazo mortal ocurre cuando dos personas tratan de poner un pie en la misma piedra. El abrazo mortal puede ser resuelto si cada una de las personas se retira hacia el lado del río desde el cual inició a cruzar. En términos de sistemas operativos, esta retirada es llamada un “**rollback**”. Observe que si varias personas están cruzando el río desde el mismo lado, es necesario que más de una persona se retire con el fin de resolver el abrazo mortal. Si una persona empieza a cruzar el río sin averiguar si alguien más está tratando de cruzar el río desde el otro lado, entonces siempre puede ocurrir un abrazo mortal.

La única manera de asegurar que un abrazo mortal no va a ocurrir, es el de pedir a cada persona que va a cruzar el río que siga un protocolo de aceptación previa. Tal protocolo debe de pedir a cada una de las personas que deseen cruzar el río, el averiguar si alguien más está cruzando desde el otro lado. Si la respuesta es no, entonces puede proseguir. De otra manera, debe esperar hasta que la otra persona haya terminado de cruzar. Varias observaciones deben de hacerse con respecto a este protocolo:

- Se debe de tener un mecanismo para determinar cuando alguien está cruzando el río. Si es posible el examinar siempre el estado de todas las piedras de apoyo, esta condición es suficiente. Si no (por ejemplo, puede ser que el río sea muy ancho o haya neblina y esté muy oscuro), otro mecanismo es necesario.
- Supongamos que dos personas quieren cruzar el río desde lados opuestos al mismo tiempo. Nuestro protocolo no especifica que debe de hacerse en este caso. Si ambos inician a cruzar el río, un abrazo mortal puede ocurrir. Si cada uno espera al otro para empezar, otra forma de abrazo mortal puede ocurrir.

Un remedio para esta dificultad es asignar a un lado del río mayor prioridad (digamos, al lado oeste). Esto es, la persona que intente cruzar desde el lado oeste, siempre cruzará primero, mientras que la persona del este tiene que esperar.

- Si este protocolo es observado, uno a más procesos pueden tener que esperar indefinidamente para cruzar el río. Esta situación es llamada "**Inanición**" (**Starvation**). Puede ocurrir, por ejemplo, que exista en flujo de gente constante desde el lado con más alta prioridad. Con el fin de prevenir la inanición, el protocolo anterior debe de ser modificado. Por ejemplo, podemos definir un algoritmo que alterne la dirección de cruce de vez en cuando.

En conclusión podemos decir que un proceso cuya solicitud de un recurso ha sido denegada entrará por lo general en un ciclo, en el cual solicita el recurso, duerme e intenta de nuevo. Aunque este proceso no está bloqueado, para todos los efectos está como bloqueado, puesto que no puede hacer ninguna labor útil.

Es así como bloqueos y/o abrazos mortales se pueden denominar o son los mismos "Interbloqueos". No importa su origen, sólo interesa que existe un proceso que está en espera de que un recurso le sea asignado.

El Interbloqueo, de forma general, se puede definir así:

Un conjunto de procesos se encuentra en estado de interbloqueo cuando cada uno de ellos espera un evento que sólo puede originar otro proceso del mismo conjunto.

En la mayoría de los casos, el evento que espera cada proceso es la liberación de cierto recurso que posee por el momento otro miembro del conjunto. En otras palabras, cada miembro del conjunto de procesos bloqueados espera un recurso poseído por un proceso bloqueado. Ninguno de los procesos puede continuar su ejecución, ni liberar recursos, y puede ser despertado.

Debe ser obvio que el interbloqueo es una condición indeseable. En un abrazo mortal o bloqueo, los procesos nunca terminan de ejecutarse y los recursos del sistema están amarrados, evitando que otros procesos puedan siquiera empezar. Antes de discutir varios métodos para manejar este problema, sería útil describir algunas de las propiedades que los caracterizan.

3.3.3 Condiciones para producir un interbloqueo¹⁸

Según Coffman (1971), existen cuatro condiciones que deben cumplirse para que haya estancamiento, bloqueo o interbloqueo. Una situación puede surgir si y solo si las siguientes cuatro condiciones ocurren simultáneamente en un sistema:

3.3.1 Exclusión mutua

Cada recurso se asigna por lo regular exactamente a un proceso o bien esta disponible. Al menos un recurso es mantenido en un modo no-compartible; esto es, sólo un proceso a la vez puede usar el recurso. Si otro proceso solicita ese recurso, tiene que ser retardado hasta que el recurso haya sido liberado.

3.3.2 Retener y esperar

Los procesos que regularmente contienen recursos otorgados antes pueden solicitar nuevos recursos. Debe existir un proceso que retenga al menos un recurso y esté esperando para adquirir recursos adicionales que están siendo retenidos por otros procesos.

3.3.3 No existe el derecho de desasignar (No preemption)

Los recursos previamente otorgados no pueden extraerse por la fuerza de un proceso. Deben ser liberados explícitamente por el proceso que los contiene. Los recursos no pueden ser desasignados (preempted); esto es, un recurso sólo puede ser liberado voluntariamente por el proceso que lo retiene, después de que el proceso ha terminado su tarea.

3.3.4 Espera circular

Debe haber una cadena de dos o más procesos, cada uno de los cuales esté esperando u recurso contenido en el siguiente miembro de la cadena. Debe existir un conjunto $\{p_0, p_1, \dots, p_n\}$ de procesos en espera tal que p_0 esté esperando por un recurso que está siendo retenido por p_1 , p_1 está esperando por un recurso que está siendo retenido por p_2, \dots, p_{n-1} y p_n está esperando por un recurso que está siendo retenido por p_0 .

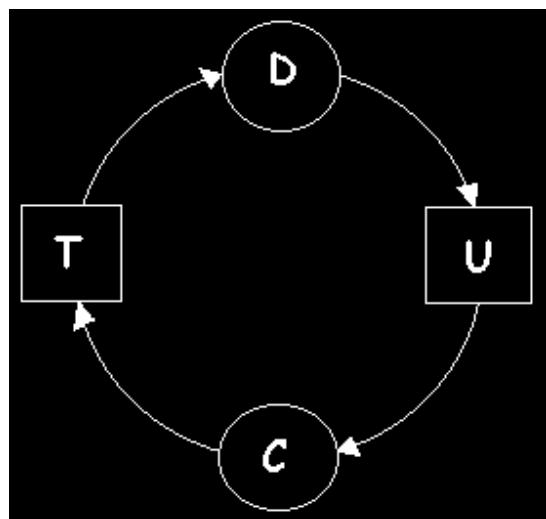
Enfatizo que las cuatro condiciones deben de cumplirse para que pueda ocurrir un abrazo mortal. La condición de espera circular implica la condición de retener y esperar, de tal manera que las cuatro condiciones no son totalmente

¹⁸La información de este tema fue consultada y extractada de los documentos Sistemas Operativos I y Sistemas Operativos II. Instituto Tecnológico de Veracruz. <http://www.itver.edu.mx/so1/> Unidad IV y <http://www.itver.edu.mx/so2/> Unidad II, respectivamente.

independientes. Sin embargo, puede ser útil el considerar cada condición por separado.

Una forma de modelar estas condiciones es usando un **grafo de recursos**: los círculos representan procesos, los cuadrados recursos. Una arista desde un recurso a un proceso indica que el recurso ha sido asignado al proceso. Una arista desde un proceso a un recurso indica que el proceso ha solicitado el recurso, y está bloqueado esperándolo. Entonces, si hacemos el grafo con todos los procesos y todos los recursos del sistema y encontramos un ciclo, los procesos en el ciclo están bajo bloqueo mutuo.

Gráfica 23. Ejemplo de un grafo de recursos



3.4 Métodos para manejar el interbloqueo

Principalmente, existen dos métodos para manejar el interbloqueo. Podemos usar algún protocolo para asegurar que el sistema nunca entrará en un estado de abrazo mortal. Alternativamente, podemos permitir que el sistema entre en un estado de abrazo mortal (bloqueo) y después recuperarse. Pero el recuperarse puede ser muy difícil y muy caro.

Por ello, en primer lugar, consideraremos los métodos para asegurar que no ocurran los abrazos mortales. Comúnmente, existen dos métodos:

- **Prevención** - Deadlock Prevention (3.5)
- **Evasión** - Deadlock Avoidance (3.6, 3.7 y 3.8)

Después se trabajarán los métodos para tratar el interbloqueo, una vez éste se ha presentado o no ha sido posible evitarlo, esto es:

- **Detección y recuperación** (3.9)

3.5 Prevención del interbloqueo¹⁹

La estrategia básica de la prevención del interbloqueo consiste, a grandes rasgos, en diseñar un sistema de manera que esté excluida, a priori, la posibilidad de interbloqueo.

Los métodos para prevenir el interbloqueo consisten en impedir la aparición de alguna de las cuatro condiciones necesarias, antes mencionadas. Al asegurarnos de que por lo menos una de estas condiciones no se presente, podemos prevenir la ocurrencia de un bloqueo.

3.5.1 Exclusión mutua

Si ningún recurso se puede asignar de forma exclusiva, no se produciría interbloqueo. Sin embargo, existen recursos para los que no es posible negar la condición de exclusión mutua, pues la propia naturaleza de los mismos obliga a que sean utilizados en exclusión mutua.

No obstante, es posible eliminar esta condición en algunos recursos. Por ejemplo, una impresora es un recurso no compatible pues si se permite que dos procesos escriban en la impresora al mismo tiempo, la salida resultará caótica. Pero con el spooling de salida varios procesos pueden generar salida al mismo tiempo. Puesto que el spooler nunca solicita otros recursos, se elimina el bloqueo originado por la impresora.

El inconveniente es que no todos los recursos pueden usarse de esta forma (por ejemplo, la tabla de procesos no se presenta al spooling y, además, la implementación de esta técnica puede introducir nuevos motivos de interbloqueo, ya que el spooling emplea una zona de disco finita).

3.5.2 Retener y esperar

Con el fin de asegurar que la condición de retener y esperar nunca ocurra en el sistema, se debe garantizar que siempre que un proceso solicite un recurso, éste no pueda retener otros recursos.

Un protocolo que puede ser usado requiere que cada proceso solicite y le sean asignados todos los recursos antes de que empiece su ejecución. Esto puede ser

¹⁹Ibid. Unidad IV y Unidad II.

implantado haciendo que las llamadas al sistema solicitando recursos se hagan antes que todas las otras llamadas al sistema.

Un protocolo alternativo permite que un proceso solicite recursos cuando no tiene ningún recurso asignado. Un proceso puede solicitar algunos recursos y utilizarlos, pero, antes de que pueda volver a pedir recursos adicionales, debe liberar todos los recursos que tiene previamente asignados.

Para ejemplificar la diferencia entre estos dos protocolos, consideremos un proceso que copia un archivo de una unidad de cinta(1) a una unidad de disco, ordena el archivo en disco, después imprime los resultados a una impresora en línea y finalmente copia el archivo de disco a la unidad de cinta(2).

Si todos los recursos deben ser solicitados al inicio del proceso, entonces el proceso debe iniciar solicitando la unidad de cinta(1), la unidad de disco, la impresora en línea y la unidad de disco(2). Este proceso mantendrá las unidades de cinta (1) y (2) durante la ejecución completa aunque sólo las utilice al principio y al final de su ejecución respectivamente.

El segundo método permite al proceso el solicitar inicialmente solo la unidad de cinta(1) y la unidad de disco. Ahora puede copiar el archivo de la unidad de cinta (1) a la unidad de disco y ordenarlo, posteriormente debe liberar la unidad de cinta(1) y la unidad de disco. El proceso debe volver a solicitar la unidad de disco y la impresora en línea para hacer la impresión. Despues de imprimir debe liberar la unidad de disco y la impresora. Ahora debe de volver a solicitar la unidad de disco y la unidad de cinta(2) para copiar el archivo a cinta, debe liberar ambos recursos y terminar.

Existen dos desventajas principales en estos protocolos:

- La primera, la utilización de los recursos puede ser muy baja, debido a que varios de los recursos pueden estar asignados pero no son utilizados la mayor parte del tiempo.
- Segundo, puede ocurrir Inanición (Starvation). Un proceso que necesite varios recursos que sean muy solicitados puede tener que esperar indefinidamente mientras al menos uno de los recursos que necesita esté asignado siempre a algún otro proceso.

En esta segunda condición, retener y esperar, si se puede impedir que los procesos que tienen los recursos esperen la llegada de más recursos podemos erradicar los estancamientos.

La solución que exige que todos los procesos soliciten todos los recursos que necesiten a un mismo tiempo y bloqueando el proceso hasta que todos los recursos puedan concederse simultáneamente resulta ineficiente por dos factores:

- En primer lugar, un proceso puede estar suspendido durante mucho tiempo, esperando que concedan todas sus solicitudes de recursos, cuando de hecho podría haber avanzado con sólo algunos de los recursos.
- Y en segundo lugar, los recursos asignados a un proceso pueden permanecer sin usarse durante períodos considerables, tiempo durante el cual se priva del acceso a otros procesos.

3.5.3 No existe el derecho de desasignar

La tercera condición necesaria es que no debe de existir el derecho de desasignar recursos que han sido previamente asignados.

Con el fin de asegurar que esta condición no se cumpla, el siguiente protocolo puede ser usado:

Si un proceso que está reteniendo algunos recursos solicita otro recurso que no puede ser asignado inmediatamente (es decir, el proceso tiene que esperar), entonces todos los recursos que tiene este proceso en espera son desasignados.

Entonces, los recursos son liberados implícitamente. Los recursos liberados son agregados a la lista de los recursos por los cuales está esperando el proceso. El proceso sólo puede volver a ser reinicializado cuando haya obtenido otra vez todos los recursos que tenía asignados, así como el nuevo recurso que estaba solicitando.

Alternativamente si un proceso solicita algunos recursos, primero verificamos si estos están disponibles. Si es así, entonces se le asignan. De otro modo, se verifica si están asignados a alguno de los procesos que están en espera de recursos adicionales. Si es así, los recursos deseados son desasignados del proceso en espera y asignados al proceso solicitante. De otra manera (no están disponibles, ni asignados a procesos en espera) el proceso que hace la solicitud debe esperar. Pero, mientras esta esperando, algunos de sus recursos pueden ser desasignados solamente si estos son solicitados. Un proceso puede ser reinicializado cuando se le asigna el recurso que había solicitado y recupera cualquier recurso que haya sido desasignado mientras esperaba.

Este protocolo es utilizado usualmente sobre recursos cuyo estado puede ser fácilmente salvado y restablecido posteriormente, ejemplo de ellos son los registros del CPU y el espacio en la memoria. Este no puede ser aplicado a recursos tales como impresoras.

Esta tercera condición (llamada también sin prioridad) es aún menos premisoria que la segunda. Si a un proceso se le ha asignado la impresora y se encuentra a

la mitad de la impresión, y tomamos a la fuerza la impresora porque no se dispone de una graficadora, se engendraría una confusión.

3.5.4 Espera circular

Con el fin de asegurarse de que la condición de espera circular nunca se presente, se puede imponer un ordenamiento total sobre todos los tipos de recursos. Esto es, asignamos a cada tipo de recurso un número entero único, el cual permita comparar dos recursos y determinar cuándo uno precede al otro en el ordenamiento.

Más formalmente, sea $R = \{r_1, r_2, \dots, r_n\}$ el conjunto de tipos de recursos. Puede definirse una función uno a uno $F:R \rightarrow N$, donde N es el conjunto de números naturales. Por ejemplo, si el conjunto R de tipos de recursos incluye unidades de disco (UDD), unidades de cinta (UDC), lectoras ópticos (LO) e impresoras (I), entonces la función F puede ser definida como sigue:

$$\begin{aligned} F(LO) &= 1 \\ F(UDD) &= 5 \\ F(UDC) &= 7 \\ F(I) &= 12 \end{aligned}$$

Se puede considerar el siguiente protocolo para prevenir interbloqueo:

Cada proceso puede solicitar recursos solamente en un orden creciente de numeración. Esto es un proceso puede solicitar inicialmente cualquier número de instancias de un tipo de recurso, digamos r_i .

Después de esto, el proceso puede solicitar instancias de recursos del tipo r_j si y solo si $F(r_j) > F(r_i)$. Si varias instancias del mismo tipo de recurso son necesitadas, debe hacerse una sola petición para todas las instancias. Por ejemplo, usando la función definida anteriormente, un proceso que quiere usar el lector óptico (LO) y la impresora (I) debe solicitar primero el (LO) y después la (I).

Alternativamente, se puede requerir simplemente que siempre que un proceso solicite una instancia del recurso tipo r_j , éste tenga que liberar cualquier recurso del tipo r_i tal que $F(r_i) \geq F(r_j)$.

Si este protocolo es usado, la condición de cola circular no puede presentarse. Se puede demostrar este hecho asumiendo que existe una cola circular (prueba por contradicción).

Sea $\{p_0, p_1, p_2, \dots, p_n\}$ el conjunto de procesos que están en la espera circular, donde p_i está esperando por un recurso r_i , el cual es retenido por

p_{i+1} . Entonces, debido a que el proceso p_{i+1} está reteniendo el recurso r_i mientras está esperando el recurso r_{i+1} , se debe tener $F(r_i) < F(r_{i+1})$, para toda i .

Pero esto significa que $F(r_0) < F(r_1) < \dots < F(r_n) < F(r_0)$. Por transitividad, $F(r_0) < F(r_0)$, lo cual es imposible. Por lo tanto, no puede existir una espera circular.

Debe notarse que la función F debe definirse de acuerdo al ordenamiento de uso normal de los recursos en el sistema. Por ejemplo, usualmente se utiliza primero una (UDD) antes que la (I), por lo cual es razonable definir $F(\text{UDD}) < F(\text{I})$.

Esta última condición (espera circular) se puede eliminar en varias formas. Una manera consiste en simplemente tener una regla que afirme que un proceso sólo tiene derechos a un recurso único en cualquier momento. Si necesita un segundo recurso debe devolver el primero.

Otra manera de evitar la espera circular es la de ofrecer una numeración global de todos los recursos. Ahora la regla es que los procesos pueden solicitar recursos siempre y cuando devuelva el recurso que tenga y el que solicite sea mayor que el que está utilizando.

Como se mencionó anteriormente el estancamiento no se evitó imponiendo reglas arbitrarias sobre procesos, sino mediante el análisis cuidadoso de cada solicitud de recursos.

3.6 Evasión del interbloqueo

En vez de restringir la forma o el orden en que los procesos deben solicitar recursos, antes se chequea que sea seguro otorgar dichos recursos. Es decir, si se presentan las condiciones suficientes para un interbloqueo, todavía es posible evitarlos por medio de una restricción en la asignación de los procesos para tratar de buscar estados seguros. Estas restricciones aseguran que al menos una de las condiciones necesarias para el interbloqueo no pueda presentarse y por lo tanto, tampoco el interbloqueo.

Otro método para evitar interbloqueo consiste en requerir información adicional sobre cómo se solicitarán los recursos. Esta información puede ser:

- La necesidad máxima de recursos de los procesos que se está ejecutando.
- La asignación actual de recursos a procesos.
- La cantidad actual de instancias libres de cada recurso.

Con base a la información que ya se tiene de la forma y del orden en que se solicitarán los recursos, se puede tomar la decisión de ejecutar el proceso o si

debe esperar. Por lo tanto, la evitación del interbloqueo sólo anticipa la posibilidad de interbloqueo y asegura que no exista nunca tal posibilidad.

3.6.1 Estados seguros e inseguros

- Un estado de asignación de recursos se considera seguro si en él no hay posibilidad de interbloqueo. Para que un estado sea seguro, es necesario que los procesos formen una secuencia segura. Una secuencia segura es una ordenación de los procesos de modo que los recursos que aún pueden pedir cualquier proceso pueden ser otorgados con los recursos libres más los recursos retenidos por los demás procesos. Con base a ello, cuando un proceso realice una solicitud de recursos, el sistema se los concederá sólo en el caso de que la solicitud mantenga al sistema en un estado seguro.
- Un estado inseguro es aquel en el que puede presentarse un interbloqueo.

3.7 Mecanismos para evitar el interbloqueo²⁰

3.7.1 Algoritmo del banquero para un sólo recurso

Este algoritmo fue ideado por Dijkstra (1965). Se creó en la forma en que un banquero trata a un grupo de clientes, a quienes les otorga líneas de crédito. El banquero sabe que no todos los clientes (4) A, B, C y D necesitarán su límite de crédito máximo de inmediato, de manera que sólo ha reservado 10 unidades en lugar de 22 (total máximo disponible) para darles servicios (con este ejemplo asumiremos que los clientes son los procesos, las unidades de crédito son los recursos y el banquero es el sistema operativo).

CLIENTES	TIENE	LIMITE MAXIMO
A	0	6
B	0	5
C	0	4
D	0	7

Reserva: 10 seguro

Los clientes emprenden sus respectivos negocios y solicitan préstamos de vez en cuando, es decir solicitan recursos. En cierto momento, la situación es como se muestra en el siguiente cuadro:

²⁰Ibid. Unidad IV y Unidad II.

CLIENTES	TIENE	LIMITE MAXIMO
A	1	6
B	1	5
C	2	4
D	4	7

Este estado es seguro, puesto que con dos unidades restantes, el banquero puede retrasar todas las solicitudes excepto la de C lo cual permite que C termine y libere sus cuatro recursos. Con cuatro unidades disponibles, el banquero puede permitir que B ó D tengan las unidades necesarias.

Consideremos lo que ocurriría si se otorgara una solicitud de B de una unidad adicional, analizando esto en el esquema anterior tendríamos este nuevo esquema:

CLIENTES	TIENE	LIMITE MAXIMO
A	1	6
B	2	5
C	2	4
D	4	7

Tendríamos un estado inseguro, ya que si todos los clientes solicitaran de pronto su máximo préstamo, el banquero no podría satisfacerlas y se tendría un bloqueo.

Un estado inseguro no tiene que llevar a un bloqueo, puesto que un cliente podría no necesitar toda su línea de crédito, pero el banquero no puede contar con ese comportamiento.

El algoritmo del banquero consiste entonces en estudiar cada solicitud al ocurrir ésta y ver si su otorgamiento conduce a un estado seguro. En caso afirmativo, se otorga la solicitud; en caso contrario, se le pospone. Para ver si un estado es seguro, el banquero verifica si tiene los recursos suficientes para satisfacer a otro cliente. En caso afirmativo, se supone que estos préstamos se le volverán a pagar; entonces verifica al siguiente cliente cercano al límite y así sucesivamente. Si en cierto momento se vuelve a pagar todos los préstamos, el estado es seguro y la solicitud original debe ser aprobada.

Desventajas del algoritmo del Banquero

El algoritmo del banquero es interesante debido a que proporciona un medio de asignar los recursos para evitar el interbloqueo. Permite proseguir a los trabajos que en una situación de prevención de interbloqueo tendría que esperar.

Sin embargo el algoritmo contiene un número de desventajas importantes que pueden hacer que un diseñador escoja otro enfoque para el problema del estancamiento, las cuales se listan a continuación:

- El algoritmo requiere que existan un número fijo de recursos asignables. Como los recursos suelen requerir servicios, bien por avería o bien por mantenimiento preventivo, no podemos contar con que el número de recursos se mantenga siempre constante.
- El algoritmo requiere que la población de usuarios se mantenga constante. Esto también es irrazonable. En los sistemas multiprogramadores actuales, la población de usuarios cambia constantemente.
- El algoritmo requiere que el banquero garantice que todas las peticiones serán concedidas dentro de un intervalo de tiempo finito. Está claro que, en sistemas reales, se necesitan garantías mucho mayores que ésta.
- El algoritmo requiere que los clientes (es decir, los trabajos) garanticen que los préstamos van a ser pagados (osea que los recursos van a ser devueltos) dentro de un intervalo de tiempo finito. También en este caso se necesitan garantías mucho mayores que ésta para sistemas reales.
- El algoritmo requiere que los usuarios indiquen sus necesidades máximas por adelantado. Al irse haciendo más dinámica la asignación de recursos resulta cada vez más difícil conocer las necesidades máximas de un usuario.

Si se tuvieran múltiples recursos y procesos, el algoritmo del banquero se usa de la misma forma que el primer caso de los clientes del banco, es decir, de las unidades disponibles se usan para ver si es posible que uno de los procesos puedan terminar, y si es así, sus recursos son utilizados para terminar otro proceso, con lo cual se van terminando, hasta dar por terminado todo el estado, y si este se logra terminar con éxito, se dice que todo el estado es seguro.

Otro ejemplo sería el siguiente, en donde las peticiones originales son:

Proc \ Rec	Impresora	U. Cinta	Archivo	Plotter	Total
A	2	3	3	1	9
B	4	1	1	2	8
C	1	4	2	4	11
Total disponibles	3	3	4	2	21

El banquero podría satisfacer los recursos en una primera vuelta de la siguiente manera:

Proc \ Rec	Impresora	U. Cinta	Archivo	Plotter	Total
A	2/0	3/2	3/2	1/0	9
B	4/3	1/1	1/1	2/2	8
C	1/0	4/0	2/1	4/0	11
Total disponibles	3	3	4	2	21

Procurando satisfacer prioritariamente a los que requieren de menos recursos para liberarlos más rápido. En una siguiente vuelta podríamos tener:

Proc \ Rec	Impresora	U. Cinta	Archivo	Plotter	Total
A	2/0/2	3/2/1	3/2/1	1/0/1	9
B	4/3/1	1/1	1/1	2/2	8
C	1/0/0	4/0/2	2/1/1	4/0/1	11
Total disponibles	3	3	4	2	21

En la tabla anterior observamos que en este punto los procesos A y B ya fueron liberados, es decir, fueron satisfechos sus requerimientos de recursos, lo cual se indica subrayando el número. Después de esto, los otros procesos siguen solicitando recursos. Ambos procesos, A y B, son los procesos **primeros liberados**.

Proc \ Rec	Impresora	U. Cinta	Archivo	Plotter	Total
A	2/0/2	3/2/1	3/2/1	1/0/1	9
B	4/3/1	1/1	1/1	2/2	8
C	1/0/0/1	4/0/2/2	2/1/1	4/0/1/2	11
Total disponibles	3	3	4	2	21

Y como al proceso C aún le quedan recursos por pedir, se tiene otra vuelta en la que se le otorgan más recursos:

Proc \ Rec	Impresora	U. Cinta	Archivo	Plotter	Total
A	2/0/2	3/2/1	3/2/1	1/0/1	9
B	4/3/1	1/1	1/1	2/2	8
C	1/0/0/1	4/0/2/2	2/1/1	4/0/1/2/1	11
Total disponibles	3	3	4	2	21

En este momento el proceso C es liberado, el cual es el **segundo liberado**.

En conclusión: Suponemos: n procesos y m recursos.

Disponible[m]	Cantidad de elementos disponibles para cada recurso.
Max[n, m]	Máximo número de elementos de cada recurso que cada proceso puede pedir.
Asignación[n,m]	Número de elementos de cada recurso actualmente asignadas a cada proceso.
Necesito[n,m]	Número de peticiones de elementos de cada recurso que cada proceso aún no ha hecho.
Estructuras de datos necesarias	

3.7.2 Algoritmo del banquero para varios recursos

El algoritmo del banquero de puede generalizar para el control de varios recursos.

En la siguiente figura se muestra el funcionamiento de este caso.

PROCESOS	IMPRESORAS	U.CINTA	ARCHIVO	PLOTTER
A	2	3	3	1
B	4	1	1	2
C	1	4	2	4

Total de recursos solicitados 28.

	IMPRESORAS	U.CINTA	ARCHIVO	PLOTTER
	3	3	4	2

Recursos disponibles

Aquí vemos 2 matrices. La primera muestra el número de recursos solicitados por el momento a cada uno de los tres procesos. La segunda muestra el número de recursos disponibles. Como en el caso de un sólo recurso, los procesos deben establecer sus necesidades totales de recursos antes de su ejecución de manera que el sistema operativo pueda calcular la siguiente matriz en cualquier instante.

Si se sigue desarrollando el ejercicio anterior, poniendo en azul los recursos solicitados y en negro la cantidad dada, la matriz va quedando:

Primera Pasada

PROCESOS	IMPRESORAS	U.CINTA	ARCHIVO	PLOTTER
A (liberado)	2,2	3,3	3,3	1,1
B	4,0	1,0	1,1	2,1
C	1,1	4,0	2,0	4,0

Recursos solicitados faltantes: 16.

Segunda Pasada

PROCESOS	IMPRESORAS	U.CINTA	ARCHIVO	PLOTTER
A (liberado)	2,2	3,3	3,3	1,1
B	4,0,3	1,0,1	1,1	2,1,1
C	1,1	4,0,2	2,0,2	4,0,1

Recursos solicitados faltantes: 6.

Tercera Pasada

PROCESOS	IMPRESORAS	U.CINTA	ARCHIVO	PLOTTER
A (liberado)	2,2	3,3	3,3	1,1
B (liberado)	4,0,3,1	1,0,1	1,1	2,1,1
C	1,1	4,0,2,2	2,0,2	4,0,1,2

Recursos solicitados faltantes: 1.

Cuarta Pasada

PROCESOS	IMPRESORAS	U.CINTA	ARCHIVO	PLOTTER
A (liberado)	2,2	3,3	3,3	1,1
B (liberado)	4,0,3,1	1,0,1	1,1	2,1,1
C (liberado)	1,1	4,0,2,2	2,0,2	4,0,1,2,1

Recursos solicitados faltantes: 0.

En el desarrollo se observa que el primer proceso en ser liberado es el **A** en la primera pasada; el **B** es liberado en la tercera, y el **C** en la pasada cuatro.

Cabe señalar que éste algoritmo del banquero es maravilloso en teoría, pero es inútil en la práctica, puesto que los procesos rara vez conocen de antemano sus necesidades máximas de recursos. Además el número de procesos no es fijo sino que varía de manera dinámica al conectarse y desconectarse los usuarios.

3.7.3 Algoritmo del aveSTRUZ

El punto de vista más simple consiste en el algoritmo del aveSTRUZ: esconder la cabeza en la arena como aveSTRUZ para pretender que no existe problema alguno.

Gráfica 24. Algoritmo del aveSTRUZ



Las distintas personas reaccionan a esta estrategia de distintas maneras. Los matemáticos la consideran totalmente inaceptable y dicen que los bloqueos se deben evitar a toda costa. Los ingenieros preguntan la frecuencia esperada del problema, la frecuencia de fallas del sistema por otras razones y la seriedad de un bloqueo. Si los bloqueos ocurren en promedio una vez cada cinco años, pero el sistema falla debido a causas de hardware, errores del compilador o del sistema operativo una vez al mes, la mayoría de los ingenieros no estarían dispuestos a pagar por el rendimiento o la conveniencia de eliminar los bloqueos.

Para aclarar más este contraste, UNIX sufre potencialmente de bloqueos que ni siquiera se detectan, puesto que se rompen de manera automática. El número total de procesos en el sistema queda determinado por el número de entradas en la tabla de procesos. Así, los espacios en la tabla de procesos son recursos finitos.

Si falla un FORK debido a que la tabla está totalmente ocupada, un punto de vista razonable para el programa que realiza el FORK es esperar un tiempo aleatorio e intentar de nuevo.

Supongamos ahora que un sistema UNIX tiene 100 espacios para procesos. Se ejecutan 10 programas, cada uno de los cuales necesita crear 12 subprocessos. Después de que cada proceso ha creado 9 procesos, los 10 procesos originales están ahora en un ciclo infinito, realizando y fallando un FORK (un bloqueo). La probabilidad de que esto ocurra es minúscula, pero podría ocurrir. ¿Deberíamos abandonar los procesos y la llamada FORK para eliminar el problema?

El máximo número de archivos abiertos está restringido, por el tamaño de la tabla de nodos i, por lo que ocurre un problema similar si dicha tabla está totalmente ocupada. El espacio de intercambio en el disco también es un recurso finito.

El punto de vista de UNIX es simplemente ignorar el problema, bajo la hipótesis de que la mayoría de los usuarios preferiría un bloqueo ocasional, en vez de una regla que restringiera a todos los usuarios a un proceso, un archivo abierto y un algo de cada cosa. Si todos los bloqueos se pudieran eliminar, no tendría que haber mucho análisis. El problema es que el precio es alto, lo cual se debe principalmente a que se pondrían restricciones inconvenientes a los procesos. Así nos enfrentamos a una desagradable contradicción entre la conveniencia y lo que es correcto y una amplia discusión sobre qué es más importante.

¿Y Usted qué opina?

3.8 Nivel de implantación de estrategias²¹

Se ha visto que ninguna de las anteriores estrategias básicas para el manejo de interbloqueos es, por sí sola, apropiada para los problemas de asignación de recursos en los sistemas operativos.

Una posibilidad es combinar las estrategias básicas, usando una estrategia óptima para cada clase de recursos del sistema. El método propuesto se basa en el concepto de que los recursos pueden dividirse en clases ordenadas jerárquicamente y para cada una se aplica la técnica más apropiada para el manejo de interbloqueos.

Para ilustrar esta técnica, consideraremos un sistema que consiste en las cuatro clases de recursos siguientes:

Recursos internos. Los recursos que utiliza el sistema, como el Bloque de Control de Proceso (PCB).



Memoria central. Memoria utilizada por un trabajo de usuario.

²¹Sistemas Operativos I. Instituto Tecnológico de Veracruz. <http://www.itver.edu.mx/so1/> Unidad IV.



Recursos de trabajos. Dispositivos asignables (impresoras) y archivos.



Espacio intercambiable. Espacio para cada trabajo de usuario en el almacenamiento secundario.



Una solución mixta para los interbloqueos en este sistema ordena a las clases en la forma expuesta y para cada una se utilizan las siguientes estrategias:

Recursos internos. Se puede usar la prevención a través de la ordenación de recursos, ya que en la ejecución no es necesario elegir entre solicitudes pendientes.



Memoria central. Pueden usarse la prevención por expropiación, ya que siempre se puede intercambiar un trabajo y expropiar la memoria central.



Recursos de trabajos. Puede utilizarse la evitación, ya que la información de los requisitos de recursos puede obtenerse de las tarjetas de control de trabajos.



Espacio intercambiable. Puede emplearse la asignación previa, pues generalmente se conocen los requisitos máximos de almacenamiento.



3.9 Detección y recuperación²²

3.9.1 Detección

A diferencia de los algoritmos vistos, para evitar interbloqueos, que se deben ejecutar cada vez que existe una solicitud, el algoritmo utilizado para detectar circularidad se puede correr siempre que sea apropiado: cada hora, una vez al día, cuando el operador note que producción se ha deteriorado o cuando se queje un usuario.

Dicho algoritmo se puede explicar utilizando las gráficas de recursos dirigidos y “reduciéndolas”. Los pasos para reducir una gráfica son los siguientes (Lane & Money, 1988):

1. Encuentre un proceso que esté utilizando un recurso y que no esté en espera de uno. Este proceso se puede eliminar de la gráfica (desconectando el vínculo que une el recurso al proceso) y los recursos se pueden devolver a la “**lista disponible**”. Esto es posible porque el proceso terminará y devolverá el recurso.
2. Encuentre un proceso que nada más espere clases de recursos que no están asignados por completo. Este proceso no contribuye al bloqueo mutuo, ya que terminará por obtener el recurso que espera, terminará su trabajo y devolverá el recurso a la “**lista disponible**”.

²² FLYNN, Ida y MCHOES, Ann (2001). Sistemas operativos. Tercera Edición. México: Editorial Thomson Learning. p. 119.

3. Vuelva al paso 1 y continúe la iteración, hasta eliminar todas las líneas que conecten recursos con procesos

Si quedan líneas, esto indicará que la solicitud del proceso en cuestión no puede satisfacerse y que existe un interbloqueo.

3.9.2 Recuperación

Una vez que se detecta el interbloqueo hay que desarmarlo y devolver el sistema a lo normal con tanta rapidez como sea posible. Existen varios algoritmos de recuperación, pero todos tienen una característica en común: requieren por lo menos una **victima**, un trabajo consumible, mismo que, al ser eliminada del interbloqueo, liberará al sistema. Por desgracia, para eliminar la víctima generalmente hay que reiniciar el trabajo desde el principio o a partir de un punto medio conveniente (Calingaert, 1982).

El primer método y más simple de recuperación, y el más drástico es terminar los trabajos que están activos en el sistema y volver a arrancarlos desde el principio.

El segundo método es terminar sólo los trabajos incluidos en el interbloqueo y solicitar a sus usuarios que los vuelvan a enviar.

El tercero es identificar qué trabajos intervienen en el interbloqueo y terminarlos uno por uno y comprobar la desaparición del bloqueo mutuo después de cada eliminación hasta resolverlo. Una vez liberado el sistema, se deja que los trabajos restantes terminen su procesamiento; luego, los trabajos detenidos se inician de nuevo desde el principio.

El cuarto procedimiento se puede poner en efecto sólo si el trabajo mantiene un registro, una instantánea de su progreso, de manera que se pueda interrumpir y después continuar sin tener que reiniciar desde el principio. La instantánea es como el descanso del ejemplo de la escalera: en vez de obligar a los que suben a volver a la parte inferior de la misma, sólo necesitan retirarse al descanso más cercano, esperar a que pasen los demás y reanudar la subida. En general, este método es el preferido para trabajos de ejecución larga, a fin de ayudarlos a tener una recuperación rápida.

Hasta ahora los cuatro procedimientos anteriores comprenden los trabajos que se encuentran en interbloqueo. Los dos métodos siguientes se concentran en trabajos que no están en interbloqueo y los recursos que conservan.

Uno de ellos, el quinto método de la lista, selecciona un trabajo no bloqueado, retira los recursos que contiene y los asigna a procesos bloqueados, de manera que pueda reanudarse la ejecución (esto deshace el interbloqueo).

El sexto método detiene los trabajos nuevos e impide que entren al sistema, lo que permite que los que no estén bloqueados sigan hasta su terminación y liberen sus recursos.

Por último, con menos trabajos en el sistema, la competencia por los recursos se reduce, de suerte que los procesos bloqueados obtienen los recursos que necesitan para ejecutarse hasta su terminación. Este método es el único aquí listado que no necesita una víctima. No se garantiza que funcione, a menos que el número de recursos disponibles exceda los necesarios en por lo menos uno de los trabajos bloqueados para ejecutar (esto es posible con recursos múltiples).

Deben considerarse varios factores al elegir la víctima para que tengan el efecto menos negativo sobre el sistema. Los más comunes son:

1. ***La prioridad del trabajo en consideración.*** Por lo general no se tocan los trabajos de alta prioridad.
2. ***El tiempo de CPU utilizado por el trabajo.*** No suelen tocarse los trabajos que están a punto de terminar.
3. ***La cantidad de tareas en que repercutiría la elección de la víctima.***

Además, los programas que trabajan con bases de datos también merecen un trato especial. Los trabajos que están modificando datos no se deben elegir para su terminación, ya que la consistencia y validez de la base de datos se pondría en peligro. Afortunadamente, los diseñadores de muchos sistemas de bases de datos han incluido mecanismos complejos de recuperación, por lo que se minimiza el daño a la base de datos si se interrumpe una transacción o se termina antes de completarse (Finkel, 1986).

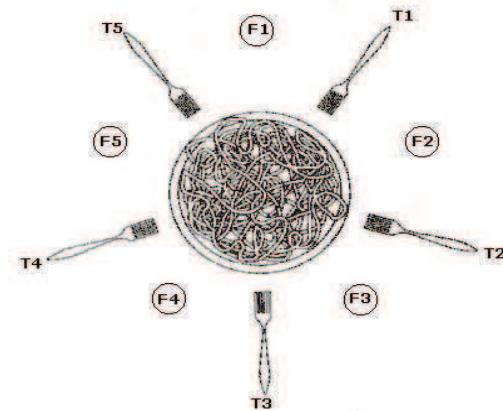
Hasta ahora hemos hablado de ***interbloqueos***, el resultado de la asignación liberal de los recursos. En el otro extremo está la ***inanición***, el resultado de la asignación de recursos, donde una tarea no puede ejecutarse porque permanece en espera de recursos que nunca quedan disponibles. Para ilustrar lo anterior, Dijkstra (1968) introdujo el caso de la “***cena de los filósofos***”.

Cinco filósofos están sentados en una mesa redonda, cada uno medita profundamente, y en el centro hay un recipiente de espagueti accesible a todos. Hay un tenedor entre cada comensal (Figura 25). La costumbre local exige que cada filósofo utilice dos tenedores, los que están a ambos lados del plato, para comer el espagueti, pero sólo hay cinco (no los diez que se requerirían para que los cinco pensadores comieran al mismo tiempo), esto es desafortunado para el filósofo 2.

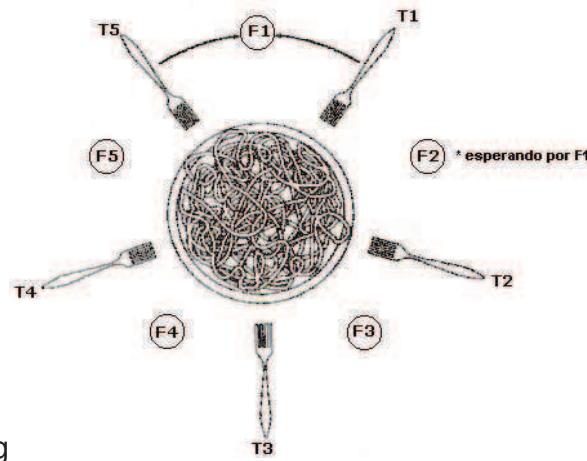
Cuando se sientan para cenar, el filósofo 1 (F1) es el primero que toma los dos tenedores (T1 y T5) a ambos lados del plato y empieza a comer. Inspirado por su

colega, el filósofo 3 (F3) hace lo mismo y toma T2 y T3. Ahora el filósofo 2 (F2) decide empezar la comida, pero no puede porque no están disponibles tenedores: T1 ha sido asignado a F1 y T2 a F3; sólo F4 o F5 puede usar el único tenedor disponible, así que el filósofo 2 debe esperar.

Gráfica 25. Cena de filósofos. Estado inicial



Pronto, F3 termina de comer, deja sobre la mesa sus dos tenedores y vuelve a su meditación. ¿Hay que asignar el tenedor que está a su lado (T2), ahora libre, al filósofo hambriento (F2)? Aunque es tentador, tal cosa resultaría ser un mal precedente, porque si se permite que los filósofos utilicen todos los recursos con la esperanza de que el recurso requerido quede disponible, la cena puede pasar con facilidad aun estado inseguro; sería cuestión de tiempo, antes que cada filósofo se hiciera de un tenedor y nadie podría cenar. Así pues, los recursos se asignan a los filósofos sólo cuando ambos tenedores están disponibles al mismo tiempo. El estado del “**sistema**”, se ilustra en la figura 26.

Gráfica 26. Cena de filósofos. Estado intermedio²³

F4 y F5 están meditando tranquilamente y F1 sigue comiendo, cuando F3 (que debe estar lleno), decide comer algo más y, dado que los recursos están libres, tiene la posibilidad de tomar T2 y T3 de nuevo. Poco después, F1 termina y libera T1 y T5, pero F2 sigue sin poder comer, porque T2 está asignado. Este escenario podría continuar de manera indefinida, y siempre que F1 y F3 alternaran su uso de los recursos disponibles, F2 deberá esperar. F1 y F3 pueden comer en cualquier momento, en tanto que F2 se muere de hambre a sólo pulgadas del alimento.

En un entorno de cómputo, los recursos son como los tenedores y los procesos competitivos son como los comensales. Si el administrador de los recursos no vigila los procesos y los trabajos que se están quedando sin alimento y planea su terminación en algún momento, se podrían quedar en el sistema esperando para que siempre la combinación correcta de recursos.

A fin de encarar este problema. Se puede implementar un algoritmo diseñado para detectar trabajos con inanición, que controla el tiempo que cada trabajo permanece en espera de recursos.

Una vez detectada la inanición, el sistema puede bloquear nuevos trabajos hasta que los trabajos hambrientos queden satisfechos. Este algoritmo se debe supervisar de cerca: si se efectúa demasiado a menudo, los nuevos trabajos quedan bloqueados constantemente y la producción disminuye. Si no se lleva a cabo con la frecuencia suficiente, quedarán en el sistema trabajos con inanición durante un periodo inaceptablemente largo.

²³ Cada filósofo debe tener ambos tenedores para empezar a comer, uno en la mano derecha y el otro en la izquierda. A menos que los recursos(tenedores) se asignen justamente, algunos filósofos pueden quedarse sin comer.

Actividad final:

Una vez desarrollada la temática de Interbloqueo y quedando claro el concepto, verifique si las situaciones reales corresponden efectivamente al concepto de interbloqueo en Sistemas Operativos. Concluya.

Realice una lista de algunos procesos que usted considera tienen interbloqueo al interior de un sistema computacional.

CAPÍTULO 4. ARQUITECTURA CLIENTE SERVIDOR

Actividad inicial:

Averigue si en su CEAD existe instalada una red de comunicaciones. Si no la hay, averigue una red en una empresa u organización de su entorno laboral. Una vez ubicado el caso averigüen cuál es la configuración y estructura que posee. Cuál es el servidor, cuáles las estaciones de trabajo, qué tipo de red es. Determine si posee una arquitectura cliente servidor.

Luego desarrolle este capítulo y concluya su estudio.

Este módulo sobre sistemas operativos, se dirige a un sistema computacional con la estructura vista en el primer capítulo, pero si bien es cierto, es importante, es aún más importante enfocarlo hacia el estudio de un sistema computacional conectado a una red de cualquier tipo, en donde exista la figura de una computadora central y varias terminales conectadas a ella. Es aquí en donde se evidencia la función, conveniencia y potencia de utilizar un sistema operativo acorde con las necesidades del sistema y en especial con las del **usuario final**.

Este capítulo está destinado a revisar los aspectos más importantes para la implementación de un sistema con arquitectura Cliente/Servidor, desde definición, consideraciones básicas de software y hardware hasta la conveniencia para los usuarios de la implantación de dicha arquitectura. Se tuvo en cuenta una parte el estudio que sobre arquitecturas y sistemas de comunicaciones presenta el Instituto Nacional de Estadística e Informática – INEI en la dirección: <http://www.inei.gob.pe/web/metodologias/attach/lib616/INDEX.HTM>

4.1 Antecedentes

Los computadores personales y los paquetes de software de aplicaciones proliferan comercialmente. Estos computadores, también conocidos como estaciones de trabajo programables, están conectados a las Redes de Área Local (LAN), mediante las cuales, los grupos de usuarios y profesionales comparten aplicaciones y datos. Las nuevas tecnologías de distribución de funciones y datos en una red, permiten desarrollar aplicaciones distribuidas de una manera transparente, de forma que múltiples procesadores de diferentes tipos (computadores personales de gama baja, media y alta, estaciones de trabajo, minicomputadoras o incluso mainframes), puedan ejecutar partes distintas de una aplicación. Si las funciones de la aplicación están diseñadas adecuadamente, se pueden mover de un procesador a otro sin modificaciones, y sin necesidad de retocar los programas que las invocan. Si se elige una adecuada infraestructura de sistemas distribuidos y de herramientas de desarrollo, las aplicaciones resultantes podrán trasladarse entre plataformas de distintos proveedores.

El desarrollo de aplicaciones Cliente/Servidor era inevitable por un conjunto de razones:

- En muchas situaciones es más eficiente que el procesamiento centralizado, dado que éste experimenta una "des-economía" de escala cuando aumenta mucho la cantidad de usuarios.
- Existían ya en ese momento servidores razonablemente eficientes y confiables.
- Se había establecido un estándar de hecho para una interface Cliente/Servidor (el ODBC SQL, adoptado por todos los fabricantes importantes de servidores).
- Era imprescindible, para apoyar con información a la creciente cantidad de ejecutivos de nivel medio que necesitan tomar decisiones ante el computador, ayudándose con las herramientas "front office", que utilizan con toda naturalidad (planillas electrónicas, procesadores de texto, graficadores, correos electrónicos, etc.).

Los primeros trabajos conocidos para la arquitectura Cliente/Servidor los hizo Sybase, que se fundó en 1984 pensando en lanzar al mercado únicamente productos para esta arquitectura. A fines de los 80's el producto fue lanzado para el voluminoso segmento "low-end" del mercado, en conjunción con Microsoft, teniendo como soporte de la base de datos un servidor OS/2, y como herramienta "front end" básica el Dbase IV de Ashton Tate. El Dbase IV no se mostró como una herramienta adecuada, y los desencuentros comerciales entre Sybase, Microsoft e IBM (en aquel momento socia de Microsoft para el OS/2) hicieron el resto.

La situación era muy diferente en 1994, cuando los principales fabricantes tradicionales (Informix, Oracle, Sybase) habían lanzado al mercado poderosos servidores y, a ellos, se agregaba IBM que estaba lanzando su producto DB2 para, prácticamente, todos los sistemas operativos importantes (además de sus clásicos MVS y VM, ahora anunciaba AIX, OS/2, Windows NT, Hewlett Packard's UNIX, Sun's UNIX, Siemens' UNIX, etc.) y Microsoft que, luego de finalizar su acuerdo con Sybase, partió para su propio SQL Server para Windows NT.

Existía un conjunto de lenguajes "front end" como, por ejemplo, Delphi, Foxpro, Powerbuilder, SQL Windows, Visual Basic, etc.

Por otra parte, en la comunidad informática existían muchas dudas sobre la calidad de los optimizadores de los sistemas de gerencia de base de datos, cuyas fallas del pasado habían sido causantes de verdaderas historias de horror.

¿Qué ha ocurrido en estos dos años?. Que los servidores se han mostrado sólidos y eficientes, que sus optimizadores probaron, en general, ser excelentes. Que una cantidad muy importante de empresas, en todo el mundo, ha encarado aplicaciones Cliente/Servidor, y quienes lo están haciendo con los planes necesarios y con las herramientas adecuadas, están obteniendo éxitos muy

importantes, mientras los que lo hicieron desaprensivamente, han cosechado fracasos.

¿Cuál es el mejor de los servidores?. Esta es una cuestión muy complicada.

Podemos tomar benchmarks publicados por cada uno de los fabricantes, o hacer los nuestros específicos, pero su importancia siempre es relativa. La respuesta, además, depende del momento en que se la formula. Para aplicaciones pequeñas y medias, todos han probado ser muy buenos, las diferencias se darán cuando se necesiten altísimos regímenes transaccionales, y dependerán de cómo cada uno vaya incorporando nuevas características como paralelismo, "read ahead", etc.

Cada nueva versión puede modificar las posiciones y los principales fabricantes están trabajando al ritmo de una gran versión nueva por año.

En general, la tecnología de los servidores de base de datos ha evolucionado mucho en los últimos años y todos los fabricantes trabajan con tecnología sensiblemente equivalente. Parecen, mucho más importantes para la elección, elementos que están fuera de la tecnología: la confianza que nos despierta el fabricante, su compromiso con el producto, su tendencia a mantenerse siempre actualizado, su situación económico/financiera, las garantías que nos brinde el soporte local y, en menor medida, el precio.

Aunque inicialmente fueron los propios usuarios quienes impulsaron esta nueva tecnología, la situación ha cambiado drásticamente. Hoy en día, el modelo Cliente/Servidor se considera intrínseco para abordar las necesidades de las empresas. El proceso distribuido se reconoce como el estándar de sistemas de información, en contraste con los sistemas independientes. Este cambio fundamental ha surgido como consecuencia de importantes factores (negocio, tecnología, proveedores), y se apoya en la existencia de una gran variedad de aplicaciones estándar y herramientas de desarrollo, fáciles de usar que soportan un entorno informático distribuido.

Inicialmente, aunque cliente servidor, se manejaba una capa (en una sola iba presentación, lógica de negocios y la estructura de datos), enseguida y hasta hace poco se manejaba el tradicional de dos capas: (una capa presentación y lógica de negocios y en la otra la estructura de datos). Y ahora, en la actualidad, se maneja el verdadero esquema cliente/servidor concebido desde un principio, con tres capas independientes, pero totalmente compatibles: **presentación, lógica de negocios y estructura datos**. Esta es la manejada con Internet (o Intranet), y todas las aplicaciones diseñadas para él.

4.2 Cliente/Servidor

El concepto de cliente/servidor proporciona una forma eficiente de utilizar todos estos recursos de máquina, de tal forma que la seguridad y fiabilidad que proporcionan los entornos mainframe se traspasa a la red de área local. A esto hay que añadir la ventaja de la potencia y simplicidad de los ordenadores personales.

La arquitectura cliente/servidor es un modelo para el desarrollo de sistemas de información, en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina **cliente** al proceso que inicia el diálogo o solicita los recursos y **servidor**, al proceso que responde a las solicitudes.

Es el modelo de interacción más común entre aplicaciones en una red. No forma parte de los conceptos de la Internet como los protocolos IP, TCP o UDP, sin embargo todos los servicios estándares de alto nivel propuestos en Internet funcionan según este modelo.

Los principales componentes del esquema cliente/servidor son entonces los clientes, los servidores y la infraestructura de comunicaciones.

En este modelo, las aplicaciones se dividen de forma que el servidor contiene la parte que debe ser compartida por varios usuarios, y en el cliente permanece sólo lo particular de cada usuario.

Los clientes interactúan con el usuario, usualmente en forma gráfica. Frecuentemente se comunican con procesos auxiliares que se encargan de establecer conexión con el servidor, enviar el pedido, recibir la respuesta, manejar las fallas y realizar actividades de sincronización y de seguridad.

Los clientes realizan generalmente funciones como:

- Manejo de la interface del usuario.
- Captura y validación de los datos de entrada.
- Generación de consultas e informes sobre las bases de datos.
-

Los servidores proporcionan un servicio al cliente y devuelven los resultados. En algunos casos existen procesos auxiliares que se encargan de recibir las solicitudes del cliente, verificar la protección, activar un proceso servidor para satisfacer el pedido, recibir su respuesta y enviarla al cliente. Además, deben manejar los interbloqueos, la recuperación ante fallas, y otros aspectos afines. Por las razones anteriores, la plataforma computacional asociada con los servidores es más poderosa que la de los clientes. Por esta razón se utilizan PCs poderosas, estaciones de trabajo, minicomputadores o sistemas grandes. Además deben

manejar servicios como administración de la red, mensajes, control y administración de la entrada al sistema ("login"), auditoría y recuperación y contabilidad. Usualmente en los servidores existe algún tipo de servicio de bases de datos. En ciertas circunstancias, este término designará a una máquina. Este será el caso si dicha máquina está dedicada a un servicio particular, por ejemplo: servidores de impresión, servidor de archivos, servidor de correo electrónico, etc.

Por su parte los servidores realizan, entre otras, las siguientes funciones:

- Gestión de periféricos compartidos.
- Control de accesos concurrentes a bases de datos compartidas.
- Enlaces de comunicaciones con otras redes de área local o extensa.
- Siempre que un cliente requiere un servicio lo solicita al servidor correspondiente y éste, le responde proporcionándolo. Normalmente, pero no necesariamente, el cliente y el servidor están ubicados en distintos procesadores. Los clientes se suelen situar en ordenadores personales y/o estaciones de trabajo y los servidores en procesadores departamentales o de grupo.

Para que los clientes y los servidores puedan comunicarse se requiere una infraestructura de comunicaciones, la cual proporciona los mecanismos básicos de direccionamiento y transporte. La mayoría de los sistemas Cliente/Servidor actuales, se basan en redes locales y por lo tanto utilizan protocolos no orientados a conexión, lo cual implica que las aplicaciones deben hacer las verificaciones. La red debe tener características adecuadas de desempeño, confiabilidad, transparencia y administración.

Entre las principales características de la arquitectura cliente / servidor, se pueden destacar las siguientes:

- El servidor presenta a todos sus clientes una interface única y bien definida.
- El cliente no necesita conocer la lógica del servidor, sólo su interface externa.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

Como ejemplos de clientes pueden citarse interfaces de usuario para enviar comandos a un servidor, APIs para el desarrollo de aplicaciones distribuidas, herramientas en el cliente para hacer acceso a servidores remotos (por ejemplo, servidores de SQL) o aplicaciones que solicitan acceso a servidores para algunos servicios.

Como ejemplos de servidores pueden citarse servidores de ventanas como X-windows, servidores de archivos como NFS, servidores para el manejo de bases

de datos (como los servidores de SQL), servidores de diseño y manufactura asistidos por computador, etc.

4.3 Componentes esenciales de la infraestructura Cliente/Servidor

Una infraestructura Cliente/Servidor consta de tres componentes esenciales, todos ellos de igual importancia y estrechamente ligados:

4.3.1 Plataforma operativa

La plataforma deberá soportar todos los modelos de distribución Cliente/Servidor, todos los servicios de comunicación, y deberá utilizar, preferentemente, componentes estándar de la industria para los servicios de distribución. Los desarrollos propios deben coexistir con las aplicaciones estándar y su integración deberá ser imperceptible para el usuario. Igualmente, podrán acomodarse programas escritos utilizando diferentes tecnologías y herramientas.

4.3.2 Entorno de desarrollo de aplicaciones

Debe elegirse después de la plataforma operativa. Aunque es conveniente evitar la proliferación de herramientas de desarrollo, se garantizará que el enlace entre éstas y el middleware no sea excesivamente rígido. Será posible utilizar diferentes herramientas para desarrollar partes de una aplicación. Un entorno de aplicación incremental, debe posibilitar la coexistencia de procesos cliente y servidor desarrollados con distintos lenguajes de programación y/o herramientas, así como utilizar distintas tecnologías (por ejemplo, lenguaje procedural, lenguaje orientado a objetos, multimedia), y que han sido puestas en explotación en distintos momentos del tiempo.

4.3.3 Gestión de sistemas

Estas funciones aumentan considerablemente el costo de una solución, pero no se pueden evitar. Siempre deben adaptarse a las necesidades de la organización, y al decidir la plataforma operativa y el entorno de desarrollo, es decir, en las primeras fases de la definición de la solución, merece la pena considerar los aspectos siguientes:

- ¿Qué necesitamos gestionar?
- ¿Dónde estarán situados los procesadores y estaciones de trabajo?
- ¿Cuántos tipos distintos se soportarán?
- ¿Qué tipo de soporte es necesario y quién lo proporciona?

Cómo definir una infraestructura Cliente/Servidor si no se acomete el trabajo de definir una infraestructura Cliente/Servidor. Se corre el riesgo de que surjan en la empresa una serie de soluciones Cliente/Servidor aisladas.

No es en absoluto recomendable el intento de una infraestructura completa desde el principio, ya que las tecnologías pueden no responder a tiempo a las necesidades prioritarias del negocio. El enfoque más adecuado está en un sistema y una plataforma de aplicación conceptuales, y una arquitectura construida incrementalmente y ampliada a medida que se desarrollan nuevas aplicaciones.

La plataforma operativa, el middleware y el entorno de desarrollo de aplicaciones están relacionados entre sí. Las necesidades de apertura pueden condicionar la elección de la plataforma o del middleware, de igual manera que lo condiciona una determinada herramienta de desarrollo. El software de aplicación puede influir en la plataforma del sistema, y el tiempo disponible para la primera aplicación puede implicar algún tipo de compromiso. Por lo tanto, es necesario fijar los objetivos y el modo de conseguirlos en cada caso concreto: una metodología de infraestructura para sistemas distribuidos que permita definir una infraestructura para el sistema Cliente/Servidor y evalúe la puesta en marcha del proyecto sobre una base racional.

El enfoque estructurado de dicha Metodología comprende los pasos siguientes:

- Captación de las necesidades. Definir, analizar y evaluar, aunando los requerimientos del negocio con las aportaciones tecnológicas.
- Diseño conceptual en el que se sitúan los principales bloques funcionales y de datos del sistema, mostrando la relación y comunicación entre ambos.
- Detalle de los principales componentes funcionales, selección de procesos, determinando los principios que deben aplicarse a la selección de software o diseño de los módulos.

Al final de los tres pasos anteriores, se definen los conceptos del sistema y la infraestructura tecnológica, sin concretar, todavía, en productos o plataformas específicos.

Por último, se llega a la selección de plataformas y principales productos y componentes para la implantación. El resultado es la descripción de una solución que incluye infraestructura tecnológica, plataformas y productos.

4.4 Características funcionales

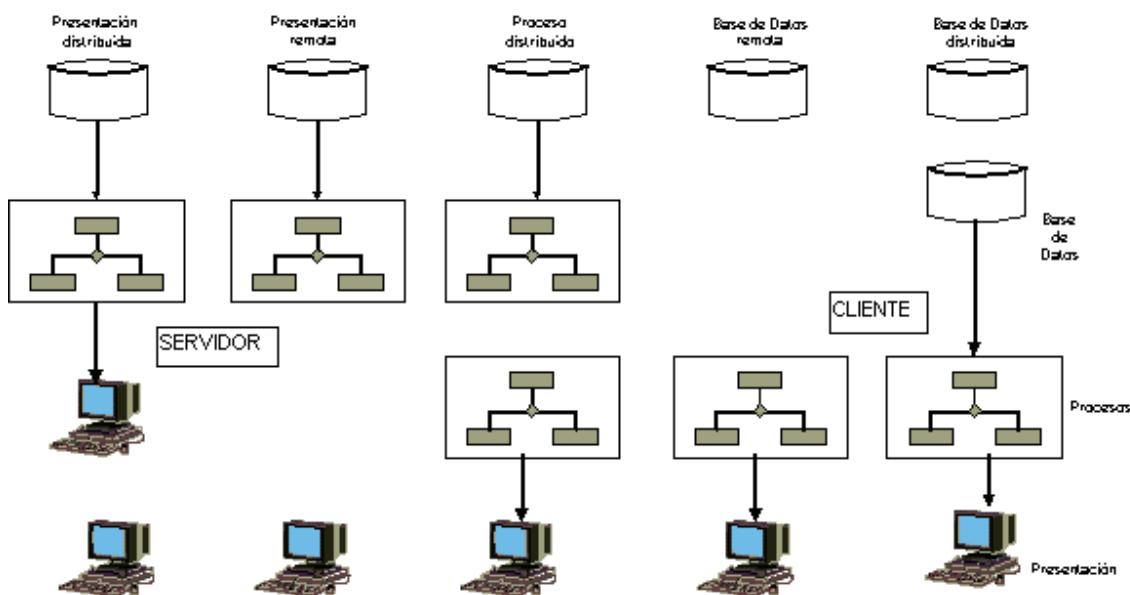
Esta arquitectura se puede clasificar en cinco niveles, según las funciones que asumen el cliente y el servidor, tal y como se puede ver en el siguiente diagrama:

- En el primer nivel el cliente asume parte de las funciones de presentación de la aplicación, ya que siguen existiendo programas en el servidor, dedicados a esta tarea. Dicha distribución se realiza mediante el uso de productos para el "maquillaje" de las pantallas del mainframe. Esta técnica no exige el cambio en las aplicaciones orientadas a terminales, pero

dificulta su mantenimiento. Además, el servidor ejecuta todos los procesos y almacena la totalidad de los datos. En este caso se dice que hay una **presentación distribuida** o embellecimiento.

- En el segundo nivel, la aplicación está soportada directamente por el servidor, excepto la presentación que es totalmente remota y reside en el cliente. Los terminales del cliente soportan la captura de datos, incluyendo una validación parcial de los mismos y una presentación de las consultas. En este caso se dice que hay una **presentación remota**.
- En el tercer nivel, la lógica de los procesos se divide entre los distintos componentes del cliente y del servidor. El diseñador de la aplicación debe definir los servicios y las interfaces del sistema de información, de forma que los papeles de cliente y servidor sean intercambiables, excepto en el control de los datos, que es responsabilidad exclusiva del servidor. En este tipo de situaciones se dice que hay un **proceso distribuido o cooperativo**.

Gráfica 27. Cinco niveles arquitectura/cliente servidor



- En el cuarto nivel el cliente realiza tanto las funciones de presentación como los procesos. Por su parte, el servidor almacena y gestiona los datos que permanecen en una base de datos centralizada. En esta situación se dice que hay una **gestión de datos remota**.

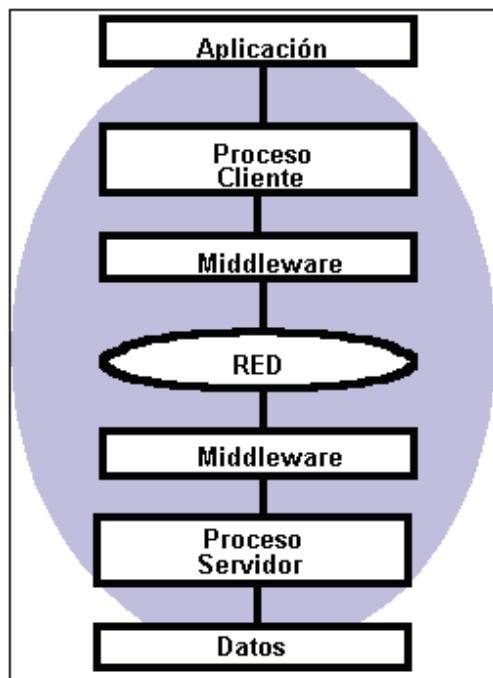
- En el quinto y último nivel, el reparto de tareas es como en el anterior y además el gestor de base de datos divide sus componentes entre el cliente y el servidor. Las interfaces entre ambos, están dentro de las funciones del gestor de datos y, por lo tanto, no tienen impacto en el desarrollo de las aplicaciones. En este nivel se da lo que se conoce como **bases de datos distribuidas**.

4.5 Características físicas

El gráfico anterior da una idea de la estructura física de conexión entre las distintas partes que componen una arquitectura cliente / servidor. La idea principal consiste en aprovechar la potencia de los ordenadores personales para realizar, sobre todo, los servicios de presentación y, según el nivel, algunos procesos o incluso algún acceso a datos locales. De esta forma se descarga al servidor de ciertas tareas para que pueda realizar otras más rápidamente.

También existe una plataforma de servidores que sustituye al ordenador central tradicional y que da servicio a los clientes autorizados. Incluso a veces el antiguo ordenador central se integra en dicha plataforma como un servidor más. Estos servidores suelen estar especializados por funciones (seguridad, cálculo, bases de datos, comunicaciones, etc.), aunque, dependiendo de las dimensiones de la instalación se pueden reunir en un servidor una o varias de estas funciones.

Gráfica 28. Distribución cliente/servidor



Las unidades de almacenamiento masivo en esta arquitectura, se caracterizan por incorporar elementos de protección que evitan la pérdida de datos y permiten multitud de accesos simultáneos (alta velocidad, niveles RAID, etc.).

Para la comunicación de todos estos elementos se emplea un sistema de red que se encarga de transmitir la información entre clientes y servidores. Físicamente consiste en un cableado (coaxial, par trenzado, fibra óptica, etc.) o en conexiones mediante señales de radio o infrarrojas, dependiendo de que la red sea local (LAN o RAL), metropolitana (MAN) o de área extensa (WAN).

Para la comunicación de los procesos con la red se emplea un tipo de equipo lógico denominado **middleware** que controla las conversaciones. Su función es independizar ambos procesos (cliente y servidor). La interface que presenta es la estándar de los servicios de red, hace que los procesos "piensen" en todo momento que se están comunicando con una red.

4.6 Características lógicas

Una de las principales aportaciones de esta arquitectura a los sistemas de información, es la interface gráfica de usuario (GUI). Gracias a ella se dispone de un manejo más fácil e intuitivo de las aplicaciones mediante el uso de un dispositivo tipo ratón. En esta arquitectura los datos se presentan, editan y validan en la parte de la aplicación cliente.

En cuanto a los datos, cabe señalar que en la arquitectura cliente / servidor se evitan las duplicidades (copias y comparaciones de datos), teniendo siempre una imagen única y correcta de los mismos, disponible en línea para su uso inmediato.

Todo esto tiene como fin que el usuario de un sistema de información soportado por una arquitectura cliente / servidor, trabaje desde su estación de trabajo con distintos datos y aplicaciones, sin importarle dónde están o dónde se ejecuta cada uno de ellos.

4.7 Middleware robusto y escalable en soluciones Cliente/Servidor

Con el paso de los años y los adelantos en la tecnología, la forma de procesar los datos dentro de las compañías y la forma de utilizar los resultados obtenidos ha tenido un constante cambio.

El éxito futuro de las compañías y su permanencia en el mercado, está directamente relacionado con la capacidad de adecuación de estas nuevas tecnologías y su correcta utilización para satisfacer las necesidades de información dentro de la empresa. En el proyecto de rediseño de la aplicación, la estrategia que se utiliza incluye el concepto de middleware.

4.7.1 Definición

El **middleware** es un módulo intermedio que actúa como conductor entre dos módulos de software. Para compartir datos, los dos módulos de software no necesitan saber cómo comunicarse entre ellos, sino cómo comunicarse con el módulo de middleware.

El middleware debe ser capaz de traducir la información de una aplicación y pasársela a la otra. El concepto es muy parecido al de ORB (Object Request Broker) que permite la comunicación entre objetos y servicios de gestión básicos para aplicaciones de objetos distribuidos.

En una aplicación cliente / servidor el middleware reside entre la aplicación cliente y la aplicación del sistema host que actúa como servidor.

El módulo middleware puede definirse también en términos de programación orientada a objetos. El módulo identifica diferentes objetos y conoce qué propiedades tienen asociadas, por lo que puede responder a peticiones referentes a los mismos.

4.7.2 Características generales

- Simplifica el proceso de desarrollo de aplicaciones.
- Es el encargado del acceso a los datos: acepta las consultas y datos recuperados directamente de la aplicación y los transmite por la red. También es responsable de enviar de vuelta a la aplicación, los datos de interés y de la generación de códigos de error.
- Es diferente desarrollar aplicaciones en un entorno middleware que la utilización de APIs (Application Programmer Interface. Interface de Programación de Aplicación) directas del sistema. El middleware debe ser capaz de manejar todas las facilidades que posee el sistema operativo y esto, no es sencillo. Por eso, muchas veces se pierde potencia con la utilización del middleware en lugar de las APIs del sistema operativo directamente.
- La adopción dentro de una organización implica la utilización de unos paquetes de software específicos para desarrollar estos módulos. Esto liga a un suministrador y a su política de actualización del producto, que puede ser distinta que la de actualización de los sistemas operativos con los que se comunica el módulo middleware.

4.7.3 Campos de aplicación

1. Migración de los sistemas host. Rediseño de aplicaciones

La aplicación debería diseñarse con base a módulos intermedios middleware, encargados de la comunicación entre el ordenador personal y el host. Esto permite desarrollar hoy la aplicación, sin tener en cuenta los futuros cambios tecnológicos que puedan sufrir los sistemas host. Si el sistema host cambia, o las aplicaciones de host se migran a plataformas de ordenadores personales, todo lo que se necesita es un nuevo módulo middleware. La interface de usuario, la lógica y el código interno permanecen sin cambios.

Por ejemplo, si el equipo lógico del sistema host se traslada desde el mainframe a una base de datos de plataforma PC ejecutándose en un servidor de ficheros, sólo hay que sustituir el módulo de middleware de forma que realice llamadas SQL.

2. Arquitectura cliente/servidor

El concepto de middleware permite también independizar los procesos cliente y servidor.

Siempre que las funciones y los objetos que se definan en el módulo intermedio middleware se basen en el flujo de actividades que realiza el usuario, éstos son válidos independientemente del entorno. Por eso, si se mantiene ese módulo separado puede servir para desarrollos futuros.

4.7.4 El middleware dentro de la empresa

El middleware es una herramienta adecuada de solución, ya que no sólo es flexible y segura, sino que también protege la inversión en tecnología y permite manejar diferentes ambientes de computación, tal como se ilustra a continuación:

Flexibilidad: La infraestructura tecnológica debe soportar crecimientos y cambios rápidos, de manera que la empresa esté en capacidad de reaccionar, de forma oportuna, en el proceso de recolección y acceso de la información importante para su funcionamiento y crecimiento. Debe estar en capacidad de adicionar nuevas soluciones en forma efectiva, eficiente y tan transparente como sea posible.

Seguridad: La infraestructura informática debe ser segura contra fallas en componentes, pérdida de información, control de acceso, entre otros. Asimismo, se necesita un nivel de seguridad, como el que brindaban los mainframes, pero en ambientes de sistemas abiertos.

Protección de la inversión y control de costos: Es importante mantener la actual inversión en tecnología. La empresa no desea desechar tecnología que

está actualmente trabajando y funcionando, así como tampoco es deseable estar constantemente haciendo reingeniería de procesos, redocumentando y reentrenando.

Diferentes ambientes de computación: Durante muchos años las organizaciones han coleccionado una serie de sistemas tipo legacy (otro nombre para identificar computadoras o sistemas con tecnología propietaria), ambientes de escritorio, soluciones Cliente/Servidor departamentales y algunas islas de información, alrededor de la empresa. Se necesita una solución que integre todas las piezas dispersas de la empresa, aumentando el acceso a la información y así permitir que la organización goce los beneficios de la computación distribuida y abierta.

Un **middleware robusto y escalable**, es la infraestructura que está en capacidad de lograr que los diversos componentes de computación de la empresa, sean vistos desde un único punto de administración. Usando un middleware adecuado, el usuario tendrá acceso seguro y confiable a la información, sabrá dónde está y cuáles son sus características, en cualquier lugar donde se tengan las siguientes condiciones:

- MS-DOS, OS/2, NT, y/o clientes windows y grandes redes tipo SNA con terminales 3270
- Servidores UNIX NCR, HP, IBM, SUN, LINUX
- Oracle, Informix, Teradata, Sybase, Ingres, ADABAS.

Adicionalmente, los desarrolladores estarán en capacidad de escribir y poner en producción rápidamente sus aplicaciones, haciendo todas las pruebas, de manera, que se garantice una perfecta distribución e implementación del nuevo módulo, para toda la empresa. El administrador podrá manejar en forma sencilla, mediante las interfaces apropiadas, todo el ambiente computacional de la compañía.

El middleware proveerá los niveles de seguridad que se necesitan, para mantener unos altos estándares de integridad de la información y una completa seguridad que la información está siendo utilizada por la persona adecuada, en la tarea adecuada.

También garantizará que los planes de contingencia que se tengan, sean viables y que se cuente con la infraestructura necesaria para colocarlos en práctica oportunamente.

Dentro de las principales características que debe cumplir un middleware que apoye a la administración de la empresa, se deben garantizar las siguientes:

- Balancear las cargas de trabajo entre los elementos de computación disponibles.

- Manejo de mensajes, que le permite entrar en el modo conversacional de un esquema Cliente/Servidor y en general, de cualquier forma de paso de mensajes entre procesos.
- Administración Global, como una sola unidad computacional lógica.
- Manejo de la consistencia entre los diferentes manejadores de bases de datos, principalmente en los procesos de OLTP (On Line Transaction Processing. Proceso transaccional en línea. Método de proceso continuo de transacciones).
- Administración de la alta disponibilidad de la solución.

¿Qué es un Middleware robusto y escalable?

Es una forma de middleware que está enfocado al manejo de aplicaciones tipo Cliente/Servidor, que coloca juntas todas las piezas de computación a través de una empresa (redes distribuidas WAN). Provee conexión sin costuras a todos sus actuales componentes de computación, junto con la posibilidad de manejar en forma centralizada un ambiente distribuido.

Este middleware debe estar en capacidad de correr en diferentes plataformas, crecer según las necesidades de la empresa y permitir la completa integración entre los diferentes niveles de computación y las herramientas que sean utilizadas. Del mismo modo, cumplir con las funciones de un monitor de transacciones.

Las soluciones que requieren de este tipo de middleware son aplicaciones que corren en forma distribuida, en múltiples y heterogéneos nodos, que accesan múltiples y heterogéneas bases de datos.

4.8 Análisis de las diferentes variantes de la arquitectura Cliente/Servidor

Existe un conjunto de variantes de la arquitectura Cliente/Servidor, dependiendo de dónde se ejecutan los diferentes elementos involucrados:

- Administración de los datos.
- Lógica de la aplicación.
- Lógica de la presentación.

4.8.1 Presentación distribuida

La primera variante que tiene algún interés es la llamada presentación distribuida, donde tanto la administración de los datos, como la lógica de la aplicación, funcionan en el servidor y la lógica de la presentación se divide entre el servidor (parte preponderante) y el cliente (donde simplemente se muestra).

Esta alternativa es extremadamente simple, porque generalmente no implica programación alguna. ¿Qué se obtiene con ella? Una mejor presentación, desde el punto de vista estrictamente cosmético, y ciertas capacidades mínimas para vincular las transacciones clásicas con el entorno Windows (un muy buen ejemplo de esta alternativa se consigue utilizando por ejemplo, el producto Rumba de Walldata).

Desde el punto de vista del uso de los recursos, esta primera alternativa es similar a la Arquitectura Centralizada.

4.8.2 Administración de datos remota

Una segunda alternativa plausible es la administración de datos remota, donde dicha administración de los datos se hace en el servidor, mientras que tanto la lógica de la aplicación, como la de la presentación, funcionan en el Cliente.

Desde el punto de vista de las necesidades de potencia de procesamiento, esta variante es la óptima. Se minimiza el costo del procesamiento en el Servidor (sólo se dedica a administrar la base de datos, no participando en la lógica de la aplicación que, cada vez, consume más recursos), mientras que se aumenta en el cliente, donde es irrelevante, teniendo en cuenta las potencias de Cliente necesarias, de todas maneras, para soportar el sistema operativo Windows.

El otro elemento a tener en cuenta es el tránsito de datos en la red. Esta variante podrá ser óptima, buena, mediocre o pésima, de acuerdo a este tránsito.

En el caso de transacciones o consultas activas, donde prácticamente todos los registros seleccionados son necesarios para configurar las pantallas a mostrar, este esquema es óptimo.

Por otro lado, en el caso de programas "batch", donde en realidad no se muestra nada, esta alternativa es teóricamente indefendible (no obstante, si el cliente está ligado al servidor por una red de alta velocidad, los resultados prácticos, a menudo, son aceptables).

Una variante interesante es la de complementar el procesamiento en el cliente con procesamiento en el servidor. Este objetivo se puede abordar de dos maneras bastante diferentes: La primera es el uso de "Stored Procedures" y "Triggers" asociados al servidor de base de datos.

4.8.3 Three Tiered Architecture

En este caso se tiene total libertad para escoger dónde se coloca la lógica de la aplicación: en el cliente, en el servidor de base de datos, o en otro(s) servidor(es). También se tiene total libertad para la elección del lenguaje a utilizar.

Se utiliza un lenguaje de tipo general (probablemente C) por lo que no existen restricciones de funcionalidad.

Los programas serán óptimos desde el punto de vista de la performance.

También deberá implementarse especialmente el Call remoto, lo que seguramente se hará de una forma más libre que los Remote Procedure Call actualmente disponibles.

No existe compromiso alguno con el uso de lenguajes propietarios, por lo que las aplicaciones serán totalmente portables sin cambio alguno.

Puede determinarse en qué servidor(es) se quiere hacer funcionar estos procedimientos. En aplicaciones críticas se pueden agregar tantos servidores de aplicación como sean necesarios, de forma simple, y sin comprometer en absoluto la integridad de la base de datos, obteniéndose una escalabilidad muy grande sin necesidad de tocar el servidor de dicha base de datos.

El problema de esta arquitectura es ¿cómo se implementa?. Parece ilusorio tratar de programar manualmente estos procedimientos, mientras que, si se dispone de una herramienta que lo hace automáticamente, presenta ventajas claras sobre la alternativa anterior:

¿Cuál será la tendencia? ¿Cuál es la mejor solución?

Hoy se está ante las primeras soluciones Three Tiered Architecture. La adopción de esta alternativa depende fundamentalmente de la disponibilidad de herramientas para generar automáticamente los procedimientos.

Se piensa que la tendencia general será una combinación adecuada entre **administración remota de datos** (que es el esquema más utilizado hoy) y **Three Tiered Architecture**.

Una pregunta que probablemente se formulará, en este esquema, ¿qué ocurre con los "triggers"? En este esquema los "triggers" siguen funcionando, de la misma forma que lo hacen en el anterior y, en vez de llamar "stored procedures" llamarán a estas rutinas C.

4.9 Arquitecturas Cliente/Servidor independientes de plataforma

¿Cómo hacer para que máquinas con arquitecturas diferentes, trabajando con sistemas operativos diferentes, con SGBD's diferentes, comunicándose con diferentes protocolos, sean capaces de integrarse entre sí?

Esta cuestión ha sido muy estudiada en las últimas dos décadas. A pesar de los avances que se han alcanzado en esta área, todavía no existe una transparencia total.

El establecimiento de patrones es una tentativa. Existen varias instituciones que son responsables en definir patrones en términos de lenguajes y sistemas, como la ANSI (American National Standards Institute) y la ISO (International Organization for Standardization).

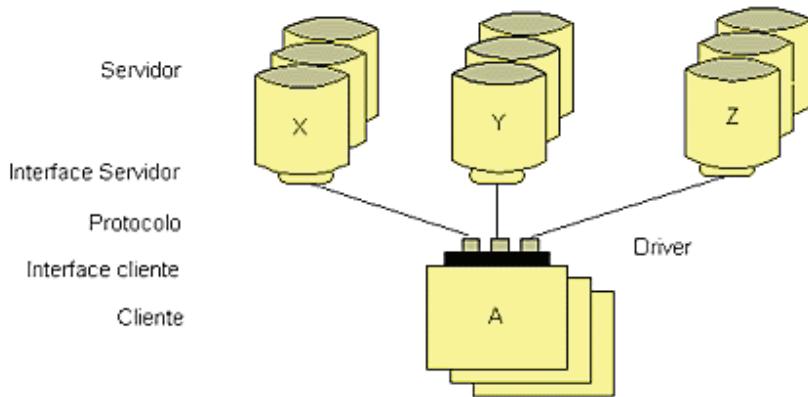
En el área de banco de datos, por ejemplo, fue creado un patrón para el SQL (Structured Query Language), que es el lenguaje más utilizado actualmente en el contexto del modelo relacional, el ANSI-SQL, como fue bautizado, sería un lenguaje de referencia a ser soportado por todos los vendedores de sistemas. Mas eso todavía no ha acontecido, en función del ANSI-SQL, es deficiente frente a extensiones de SQL, éstos, producidos por vendedores que incorporan nuevas características de un SGBD como patrón, ganando performance y ventaja competitiva frente a sus competidores.

Así, ahora, bajo un patrón SQL, las extensiones de SQL de cada fabricante que generalmente exploran mejor las cualidades de un servidor de banco de datos, son una ventaja del punto de vista de desempeño. Por otro lado, es una desventaja la pérdida de portabilidad. Las opciones generalmente consideradas son: la utilización de un conjunto de instrucciones que sean comunes a todos los SQL o la utilización de los llamados drivers.

El uso de drivers ha sido otra tentativa de mitigar la cuestión de transparencia y de explorar mejor estos avances tecnológicos, todavía no incorporados como patrones. Los drivers son programas complejos con el conocimiento específico de una determinada máquina y/o sistema. Ellos realizan la traducción de los requisitos de una interface genérica (sobre el cual un aplicativo fue construido) para un sistema específico, y viceversa.

Con o sin la ayuda de drivers, existen tres formas para implementar una arquitectura cliente/servidor de banco de datos, que puedan ser independientes de la plataforma: interface común, gateway común, y protocolo común. Todas ellas se basan en el uso de un traductor como un elemento común que irá a efectuar las transacciones de aplicación con un SGBD.

La forma **interface común**, utiliza una interface de cliente como traductor. Eso implica que la aplicación se deba preocupar solamente con la interface de cliente, que sería la responsable de la comunicación con el servidor. Generalmente, ella cuenta con el auxilio de drivers para optimización de contacto con moldes de protocolo y la interface de servidor (Figura 1).

Gráfica 29. Arquitectura cliente/servidor con interface común

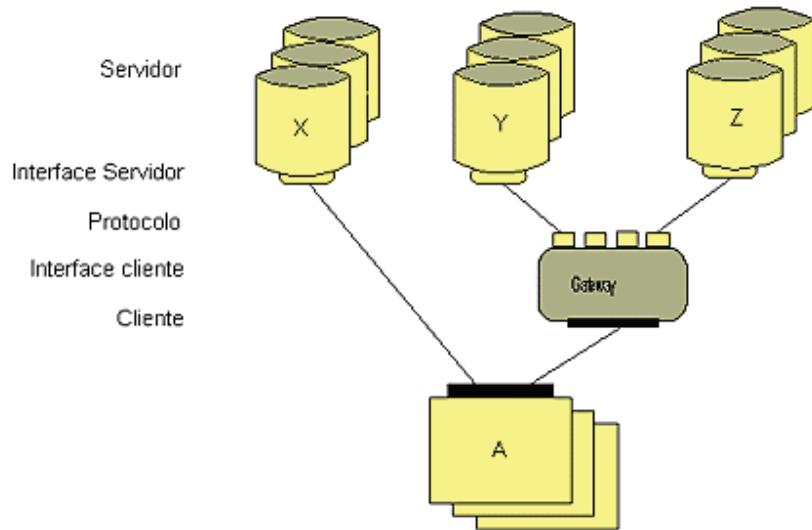
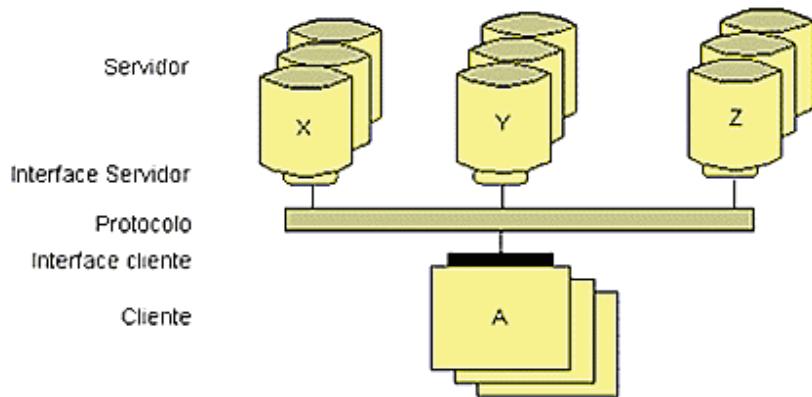
Una forma gateway²⁴ común, como dice su nombre, usa una pasarela común (es un sistema altamente comprometido con la comunicación) como traductor.

Normalmente, un gateway localiza una plataforma separada de plataformas de cliente y servidor. Siendo un sistema especializado en traducción, el gateway ofrece grandes cantidades de drivers para diversos tipos de protocolos de interfaces cliente y de interfaces servidor. (Gráfica 30).

Por último, la forma protocolo común, utiliza un protocolo común y abierto como elemento traductor. Esta forma no necesariamente implica el uso de drivers, ya que basta que ambas interfaces, cliente y servidor, entiendan el mismo protocolo. (Gráfica 31).

Ninguna de las tres, por sí solas, resuelve convenientemente el problema de transparencia entre plataformas. En verdad, una implementación práctica ha sido una combinación de estas tres formas.

²⁴ Puerta de acceso, pasarela. Unidad de interfuncionamiento. Dispositivo de comunicaciones que interconecta sistemas diseñados conforme a protocolos propietarios, o entre un sistema con un protocolo propietario y un sistema abierto o una red RAL, teniendo lugar una conversión completa de protocolos hasta la capa 7 del modelo de referencia OSI.

Gráfica 30. Arquitectura cliente/servidor con gateway común**Gráfica 31. Arquitectura cliente/servidor con protocolo común**

4.10 Condiciones para la implantación del modelo Cliente/Servidor

Las condiciones que pueden aconsejar la implantación del modelo Cliente/Servidor en una empresa son:

- Cambios estructurales y organizativos.
- Cambios en los organigramas, con mayor delegación en personas y departamentos.

- Respuesta a la dinámica del mercado.
- Cambios en los procesos de negocio.

La situación está cambiando. De una época anterior de masiva producción industrial, estamos pasando a otra de ajustada adaptación a la demanda. La capacidad de aproximación de los productos y servicios, a la medida de las necesidades del cliente, exige diseñarlos, producirlos y suministrarlos con rapidez y mínimos costos.

Las razones que impulsan el crecimiento de las aplicaciones Cliente/Servidor son:

- La demanda de sistemas más fáciles de usar, que contribuyan a una mayor productividad y calidad.
- El precio/rendimiento de las estaciones de trabajo y de los servidores.
- La creciente necesidad de acceso a la información para tomar decisiones y de soportar los procesos mediante unas aplicaciones más ajustadas a la estructura organizativa de la empresa, que permitan realizar las operaciones de forma más natural.
- La utilización de nuevas tecnologías y herramientas de alta productividad, más aptas para la dinámica del mercado.

4.11 Costos y beneficios de Cliente/Servidor

Los costos de la implantación de soluciones Cliente/Servidor no deben contemplarse sólo en términos absolutos, sino que deben medirse en función del beneficio que reporten los nuevos desarrollos. Como el precio de los ordenadores personales ha bajado tanto en los últimos años, con frecuencia se comete el error de pensar que las soluciones Cliente/Servidor son más económicas que las basadas en ordenadores tradicionales. Estudios independientes indican que el costo del hardware y software, en un periodo de 5 años, representa solamente el 20% de los costos totales. En el caso de sistemas distribuidos, el 80% restante son costos de infraestructura y gastos de explotación.

Los beneficios percibidos de la implantación de un modelo Cliente/Servidor se encuadran, generalmente, en alguna de estas categorías:

- La productividad que se obtiene en las estaciones de trabajo programables con interface gráfica de usuario, que permite acceder e integrar aplicaciones muy intuitivamente.
- La abundancia de software disponible comercialmente, como por ejemplo procesadores de textos, hojas de cálculo, sistemas basados en el conocimiento, correo, etc.
- La cercanía del usuario a aplicaciones y datos que son necesarios para su actividad, compartiendo servicios y costos.

- La disponibilidad de potencia de cálculo a nivel personal, sin la responsabilidad del mantenimiento del sistema y del software de aplicaciones.
- La disponibilidad de herramientas de desarrollo fáciles de usar, reduciendo la dependencia del departamento informático.

Los beneficios obtenidos por la alta dirección, seguramente estarán entre los siguientes:

- Un mejor ajuste del sistema de información a la organización y a los procesos de negocio.
- Cada tarea se puede ubicar en la plataforma que sea más eficaz, y los recursos informáticos se pueden aplicar progresivamente. Las funciones y los datos se pueden localizar donde sean necesarios para la operativa diaria sin cambiar las aplicaciones.
- Mayor protección de activos informáticos e integración de los sistemas y aplicaciones ya existentes.
- Acceso a la información cuándo y dónde la necesitan los usuarios. Este es, probablemente, el mayor activo corporativo.
- Disponibilidad de aplicaciones estándar (comprar en vez de desarrollar).
- Libertad para migrar a plataformas de sistemas alternativos y usar servidores especializados.
- Una respuesta más rápida a las necesidades del negocio gracias a la disponibilidad de software de productividad personal y de herramientas de desarrollo fáciles de usar.
- Un entorno de utilización más sencillo, que proporciona una mayor productividad. A largo plazo, las interfaces gráficas de usuario reducen los costos asociados a educación y formación de los usuarios.

Los beneficios que se derivan de una aplicación Cliente/Servidor dependen, en gran medida, de las necesidades del negocio. Por ejemplo, renovar una aplicación existente añadiéndole una interface gráfica de usuario, podría ser una solución rentable en un determinado momento, pero puede no serlo en otro.

Sin embargo, una nueva aplicación basada en un proceso de negocio rediseñado, seguramente reportará mayor beneficio que migrar a una aplicación ya existente.

Como la mayoría de las empresas han realizado una importante inversión en soluciones informáticas, muy raramente se pueden construir nuevos sistemas abandonando los ya existentes. Las inversiones se contemplan generalmente en términos de hardware instalado, pero es probable que sean más importantes las inversiones realizadas en:

- Software de sistemas.
- Datos disponibles en la empresa.

- Aplicaciones.
- Conocimientos en el departamento de SI.
- Conocimientos de los usuarios.

Todos éstos son activos fundamentales que, frecuentemente, se olvidan o se desprecian.

Cliente/Servidor posibilita una migración paso a paso, es decir, la generación de nuevas aplicaciones que coexisten con las ya existentes, protegiendo, de esta forma, todas las inversiones realizadas por la empresa.

4.12 Fases de implantación

Una arquitectura cliente / servidor debe mostrar los sistemas de información no como un cliente que accede a un servidor corporativo, sino como un entorno que ofrece acceso a una colección de servicios. Para llegar a este estado pueden distinguirse las siguientes fases de evolución del sistema:

4.12.1 Fase de iniciación

Esta etapa se centra sobre todo en la distribución física de los componentes entre plataformas. Los dos tipos de plataforma son:

- Una plataforma cliente para la presentación (generalmente un ordenador personal de sobremesa).
- Una plataforma servidora (como por ejemplo el servidor de una base de datos relacional) para la ejecución de procesos y la gestión de los datos.

Un ejemplo sería el de una herramienta de consulta que reside en un ordenador personal a modo de cliente y que genera peticiones de datos que van a través de la red hasta el servidor de base de datos. Estas peticiones se procesan, dando como resultado un conjunto de datos que se devuelven al cliente.

En esta fase pueden surgir los siguientes problemas:

- Cómo repartir la lógica de la aplicación entre las plataformas cliente y servidor de la forma más conveniente.
- Cómo gestionar la arquitectura para que permita que cualquier cliente se conecte con cualquier servidor.

4.12.2 Fase de proliferación

La segunda etapa de una arquitectura cliente / servidor se caracteriza por la proliferación de plataformas clientes y servidoras. Ahora, el entorno para la

interacción entre clientes y servidores se hace mucho más complejo. Puede hacerse una distinción entre:

- Datos de servidores a los que se accede a través de una red de área extensa (conocida como WAN) y
- Datos a los que se accede a través de una red de área local (conocida como LAN).

Los mecanismos de conexión son muy variados y suelen ser incompatibles.

En esta fase los problemas que se pueden plantear son:

- La gestión de accesos se convierte en crítica y compleja, debido a la estructura del organismo donde se está implantando la arquitectura. El mercado ofrece algunas soluciones que mejoran la interoperabilidad y que se basan en conexiones modulares que utilizan, entre otros:
 - Drivers en la parte cliente.
 - Pasarelas (gateways) a bases de datos.
 - Especificaciones de protocolos cliente / servidor, etc.
- Los requisitos de actualización de datos pasan a formar parte de los requisitos solicitados al sistema cliente / servidor. Ahora no sólo se consultan datos, sino que se envían peticiones para actualizar, insertar y borrar datos.

4.12.3 Fase de control

En esta fase se consolidan los caminos de acceso desde una plataforma cliente particular, a una plataforma servidora particular.

Los conceptos en los que se debe poner especial énfasis son los siguientes:

- **Transparencia en la localización.** Significa que la aplicación cliente no necesita saber nada acerca de la localización (física o lógica) de los datos o los procesos. La localización de los recursos debe estar gestionada por servidores y estar representada en las plataformas adecuadas, de forma que se facilite su uso por parte de las plataformas cliente.
- **Gestión de copias.** El sistema se debe configurar de forma que se permita copiar la información (datos o procesos) de los servidores.
- **Especialización de los equipos servidores**, en servidores de bases de datos o en servidores de aplicaciones. Los servidores de bases de datos

continúan ofreciendo servicios orientados a datos a través de llamadas SQL o a través de procedimientos almacenados. En cualquier caso, los servicios se orientan a mantener la integridad de los datos. Por otro lado, los servidores de aplicaciones se centran en los procesos, implementando partes de la lógica de la aplicación en la parte servidora.

4.12.4 Fase de integración

Esta etapa se caracteriza por el papel conjunto que juegan la gestión de accesos, la gestión de copias y la gestión de recursos. La gestión de la información se debe realizar de forma que se pueda entregar la información controlada por los servidores que contienen los datos a las plataformas clientes que los requieran. El concepto en que se basa este tipo de gestión es la distinción entre dos tipos de datos: datos de operación y datos de información. Para ajustarse a los posibles cambios en los procesos, los datos de operación varían continuamente, mientras que los datos de información son invariables porque son de naturaleza histórica y se obtienen tomando muestras en el tiempo, de los datos de operación.

4.12.5 Fase de madurez

Esta es la etapa final de una arquitectura cliente / servidor. Se caracteriza por una visión más flexible de las plataformas físicas del sistema que se contemplan como una única unidad lógica. Este estado también se caracteriza porque la tecnología cliente / servidor se ha generalizado en la empresa. Ya no es un problema saber qué componentes se distribuyen en qué plataformas, porque los recursos se pueden redistribuir para equilibrar la carga de trabajo y para compartir los recursos de información. Lo fundamental aquí, es saber quién ofrece qué servicios. Para ello es necesario distinguir qué tipo de servicios y recursos se demandan y conocer las características de esta arquitectura basada en servicios.

En la fase de integración veíamos que se establecía una distinción entre datos de operación y datos de información histórica. Por contra, en un entorno de operación cliente / servidor que se encuentre en la fase de madurez, lo interesante es distinguir entre dos nuevos términos: organismo y grupo de trabajo. Esta distinción se establece basándose en sus diferencias organizativas. El grupo de trabajo es el entorno en el que grupos organizados de personas se centran en tareas específicas de la actividad del organismo al que pertenecen. Estos equipos de personas requieren una información propia y unas reglas de trabajo particulares, que pueden ser diferentes de las del organismo en su globalidad.

Una arquitectura basada en servicios es la que se contempla como una colección de consumidores de servicios poco relacionados entre sí y los productores de dichos servicios. La utilización de este tipo de arquitectura permite pensar en nuevos retos de diseño:

- Desarrollo de componentes reutilizables entre distintas aplicaciones y distintos grupos de trabajo
- Desarrollo de aplicaciones distribuidas
- Gestión del desarrollo de aplicaciones entre distintos equipos, etc.

Actividad de refuerzo:

Continué con su estudio del sistema de red seleccionado.

Prepare un pequeño informe en donde muestre los resultados obtenidos de su investigación.

Socialícelo con el tutor y aplique las observaciones recibidas.

Saque una lista de las preguntas que tiene acerca del tema y aproveche al tutor para que le ayude a obtener la solución.

BIBLIOGRAFÍA

ALCALDE, Eduardo. MORERA, Juan. PEREZ-CAMPANERO, Juan A. (1994). *Introducción a los Sistemas Operativos. Serie Informática de Gestión.* México: Editorial Mc Graw Hill.

BARRETO ROA, Julio Humberto. (2001). *Sistemas Operativos. Guía de estudio.* Bogotá: Editorial UNAD.

CALDERA (2003). *Kit de Recursos. Unifying Unix Whit Linux For Business.*

CAÑAS, Javier. Documento pdf: Sistemas Operativos. Catorce capítulos (1999).

CARRETERO PEREZ, Jesús, GARCIA CABALLEIRA, Félix, ANASAGASTI, Pedro de Miguel, PEREZ COSTOYA, Fernando (2001). *Sistemas Operativos. Una visión aplicada.* Madrid: Mc Graw Hill.

FLYNN, Ida M, MCCHOES, Ann McIver.(2001) *Sistemas operativos. Tercera Edición.* México: Editorial Thomson Learning.

RAYA, Laura, ALVAREZ, Raquel, RODRIGO, Víctor. (2005). *Sistema Operativos en entornos Monousuario y Multiusuario.* México: Alfaomega, Ra-Ma.

RUEDA, Francisco. (1989). *Sistemas Operativos.* Santafé de Bogotá: Mc Graw Hill.

SILBERSCHATZ, Avi, GALVIN, Peter, GAGNE, Greg. (2002). *Sistemas Operativos.* México: Editorial Limusa Wiley.

STALLING, William. (2001). *Sistemas operativos. Cuarta edición,* México: Prentice Hall.

TACKETT, J. (2003). *Edición especial Linux.* México: Prentice Hall

TANENBAUM, S. Andrew, WOODHULL, Albert S. (1997). *Sistemas Operativos. Diseño e implementación.* México: Prentice Hall.

DIRECCIONES WEB

<http://www.tau.org.ar/base/lara.pue.udlap.mx/sistoper/>

<http://www.itver.edu.mx/so1/>

<http://www.itver.edu.mx/so2/>

<http://os.matiu.com.ar/>

<http://os-matiu.dreamhost.com/classes/clase1.html>

<http://www.iespana.es/canalhanoi/so/>

http://server2.southlink.com.ar/vap/sistemas_operativos.htm

<http://www.inei.gob.pe/web/metodologias/attach/lib616/INDEX.HTM>

<http://www.itq.edu.mx/vidatec/maestros/sis/mnogues/Unidad1.htm>

<http://www.cs.virginia.edu/~knabe/iic2332/notes01.html>

UNIDAD DIDÁCTICA 2

ADMINISTRACIÓN DE RECURSOS

INTRODUCCIÓN

Una de las principales funciones de un sistema operativo es la gestión, control y organización de los recursos disponibles en un sistema de cómputo.

Esos recursos hacen referencia, principalmente, a los procesos, memoria, dispositivos y archivos que son la base fundamental para las operaciones y acciones que ejecuta el usuario con el sistema y viceversa.

Es importante revisar cuáles son los algoritmos, pasos o tareas que debe realizar el SO, ante las diferentes acciones que se presentan, y más que acciones a los posibles errores que se puedan presentar debido a la característica que todo sistema persigue: la multitarea y la multiprogramación.

Además es importante reconocer con qué cuenta el sistema de cómputo y cuál es el máximo provecho (en tiempo, en dinero, en general en recursos) que se le puede extraer para lograr los resultados deseados y más que deseados exitosos.

En el corazón de los sistemas operativos

OBJETIVOS

1. Estudiar en detalle el concepto de proceso, la información asociada al mismo, sus posibles estados, las señales y temporizadores que pueden estar asignadas a un proceso.
2. Identificar los requisitos de la gestión de la memoria, el modelo de memoria de un proceso, diversos esquemas de gestión, incluyendo la memoria real y la virtual.
3. Establecer los parámetros y características fundamentales de los diferentes dispositivos de entrada/salida y el nivel de gestión que realiza el SO para su adecuado funcionamiento.
4. Proporcionar mecanismos para comprender el sistema de gestión de archivos y directorios por parte de un sistema operativo.
5. Conocer los diferentes mecanismos de protección entre los distintos procesos que se ejecutan en un sistema y entre los distintos sistemas que estén conectados entre sí.

CAPÍTULO 1. ADMINISTRACIÓN DE LOS PROCESOS²⁵

Actividad inicial:

Haga una lista con los que Usted considera son procesos que maneja un sistema operativo. El cuadro debe tener dos columnas: El proceso y la descripción del proceso.

Guarde la lista para la actividad final.

1.1 Introducción a los procesos

Los sistemas operativos de la antigüedad sólo permitían ejecutar un programa a la vez. Este programa obtenía el control completo del sistema.

Los sistemas operativos actuales permiten la ejecución concurrente de múltiples programas cargados en memoria. Entonces nace el concepto de **proceso**.

Un proceso es un programa en ejecución. Es la unidad de trabajo de un S.O moderno.

Un S.O se puede ver como un conjunto de procesos. De esta manera decimos que los procesos del S.O. ejecutan código de sistema y los procesos de usuarios ejecutan código de usuario.

Todos los procesos se ejecutan en forma **pseudo-concurrente**, con la CPU comutando entre ellos. De esta manera se logra que el sistema computacional sea más productivo. Veamos entonces cómo es la gestión de los procesos, por parte de un sistema operativo.

1.1.1 Concepto de proceso

Uno de los conceptos más importantes que gira entorno a un sistema operativo es el de proceso.

En términos simples, un proceso es un programa en ejecución junto con el entorno asociado (registros, variables, etc.). La ejecución de un proceso se realiza de una forma secuencial.

Los conceptos de job (tareas) y procesos son equivalentes y se pueden intercambiar. Un proceso tiene recursos como:

- Código ejecutable

²⁵ Las figuras de este capítulo fueron tomadas del documento pdf: Sistemas Operativos. Profesor Javier Cañas. Capítulos 4, 5 y 6.

- Datos
- Registros temporales
- Stack
- Program Counter

El corazón de un sistema operativo es el **núcleo**, un programa de control que reacciona ante cualquier interrupción de eventos externos y que da servicio a los procesos, creándolos, terminándolos y respondiendo a cualquier petición de servicio por parte de los mismos.

Un proceso es una actividad que se apoya en datos, recursos, un estado en cada momento y un programa.

También es importante considerar que si dos o más procesos forman parte de un mismo programa, se consideran secuencias separadas de ejecución y que pueden cooperar entre ellos.

1.1.2 Estados de un proceso

Es necesario resaltar que un proceso tiene una naturaleza dinámica, es decir es una entidad activa.

Cuando un proceso se ejecuta, cambia de estado.

Los estados de los procesos son internos del sistema operativo y transparentes para el usuario. Para éste, su proceso estará siempre en ejecución independientemente del estado en que se encuentre internamente en el sistema.

Un proceso puede estar en alguno de los siguientes estados:

- **Nuevo:** se está creando.
- **Corriendo (Run):** se están ejecutando instrucciones. El proceso tiene el control del procesador. En un sistema monoprocesador este estado sólo lo puede tener un proceso. (Ejecución)
- **Espera (Wait):** el proceso espera que ocurra algún evento. Por ejemplo el término de una operación de E/S o la recepción de una señal. Son los procesos que no pueden ejecutarse de momento por necesitar algún recurso no disponible (generalmente recursos de E/S). (Bloqueado)
- **Listo (Ready):** El proceso está listo para ocupar la CPU. Aquellos procesos que están dispuestos para ser ejecutados, pero no están en ejecución por alguna causa (interrupción, haber entrado en cola estando otro proceso en ejecución, etc). (Preparado)

- **Fin:** el proceso terminó su ejecución

Gráfica 32. Estados de un proceso

Estados de un Proceso



1.1.3 Transiciones de estados

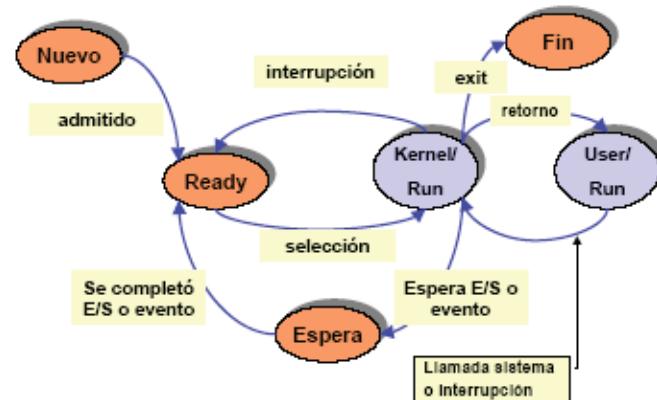
Todo proceso a lo largo de su existencia puede cambiar de estado varias veces. Cada uno de estos cambios se denomina **transición de estado**. Estas transiciones son las siguientes:

- **Comienzo de la ejecución.** Todo proceso comienza al ser dada la orden de ejecución del programa insertándose en la cola de preparados. El encolamiento dependerá de la política de gestión de dicha cola.
- **Paso a estado de ejecución.** Cuando el procesador se encuentra inactivo y en la cola de preparados exista algún proceso en espera de ser ejecutado, se pondrá en ejecución el primero de ellos.
- **Paso a estado bloqueado.** Un proceso que se encuentre en ejecución y que solicite una operación a un dispositivo externo, teniendo que esperar a que dicha operación finalice, será pasado de estado de ejecución a estado bloqueado insertándose su PCB en la cola correspondiente de bloqueados. A partir de este momento el procesador pone en ejecución el siguiente proceso, que será el primero de la cola de preparados.
- **Paso a estado preparado.** Este paso puede ser producido por alguna de las siguientes causas:

- Orden de ejecución de un programa, con lo cual el proceso pasa a la cola de preparados.
 - Si un proceso está en estado bloqueado por causa de una operación de entrada/salida y ésta finaliza, pasará de la cola de bloqueados a la de preparados.
 - Si un proceso está en ejecución y aparece una interrupción que fuerza al sistema operativo a ejecutar otro proceso, el primero pasará al estado preparado y su PCB a la cola de preparados.
 - Activación. Un proceso suspendido previamente sin estar bloqueado pasará al estado preparado al ser activado nuevamente.
- **Paso a estado suspendido bloqueado.** Si un proceso está bloqueado y el sistema operativo recibe la orden de suspenderlo, su PCB entrará en la cola de procesos suspendidos bloqueados.
- **Paso a estado suspendido preparado.** Este paso se puede producir bajo tres circunstancias:
- Suspensión de un proceso preparado pasando éste de la cola de procesos preparados a la de suspendidos preparados.
 - Suspensión de un proceso en ejecución, con lo cual el proceso pasa a la cola de suspendidos preparados.
 - Desbloqueo de un proceso suspendido bloqueado por desaparecer la causa que impedía el ser activado de nuevo.

Gráfica 33. Estados de un proceso: Modos kernel y usuario

Estados de un Proceso: Modos kernel y usuario



1.1.4 Operaciones sobre procesos

Los procesos en un S.O. pueden ejecutarse concurrentemente y deben ser creados y eliminados dinámicamente. Para esto se deben proveer llamadas al sistema que permitan:

- Crear procesos
- Destruir procesos
- Terminar procesos

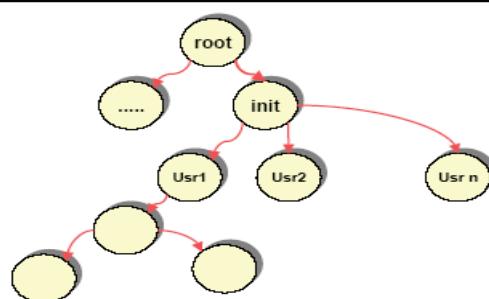
Los sistemas operativos actuales poseen una serie de funciones cuyo objetivo es el de la manipulación de los procesos. Las operaciones que se pueden hacer sobre un proceso son las siguientes:

- **Crear el proceso.** Se produce con la orden de ejecución del programa y suele necesitar varios argumentos, como el nombre y la prioridad del proceso. Aparece en este momento el PCB, que será insertado en la cola de procesos preparados. La creación de un proceso puede ser de dos tipos:
 - **Jerárquica.** En ella, cada proceso que se crea es hijo del proceso creador y hereda el entorno de ejecución de su padre. El primer proceso que ejecuta un usuario será hijo del intérprete de comandos con el que interactúa.
 - **No jerárquica.** Cada proceso creado por otro proceso se ejecuta independientemente de su creador con un entorno diferente. Es un tipo de creación que no suele darse en los sistemas operativos actuales.

Un proceso puede crear nuevos procesos. El proceso que crea se denomina **proceso padre** y los procesos creados, **procesos hijos**. Cada uno de estos procesos, puede a su vez crear nuevos procesos. De esta forma se logra una jerarquía de procesos.

Gráfica 34. Jerarquía de procesos

Ejemplo de Jerarquía de Procesos



Como cada proceso necesita recursos, éstos los puede obtener directamente del S.O, o compartir recursos con su padre.

Cuando se crea un nuevo proceso existen dos alternativas:

- El padre continúa ejecutándose en forma concurrente con el hijo.
- El padre espera hasta que alguno o todos sus hijos terminen.

- **Destruir un proceso.** Se trata de la orden de eliminación del proceso con la cual el sistema operativo destruye su PCB.

Un proceso termina cuando ejecuta su última sentencia y pide al S.O que lo elimine. Cuando esto ocurre, todos los recursos son devueltos al S.O.

Pero ¿*cuándo un proceso termina?*

- Se ejecutó la última sentencia.
- El proceso decide terminar.
- Un proceso decide matar a otro.
- Un proceso padre puede matar a sus hijos.

- **Suspender un proceso.** Es un proceso de alta prioridad que paraliza un proceso que puede ser reanudado posteriormente. Suele utilizarse en ocasiones de mal funcionamiento o sobrecarga del sistema.

- **Reanudar un proceso.** Trata de activar un proceso que ha sido previamente suspendido.

- **Cambiar la prioridad de un proceso.**

- **Temporizar la ejecución de un proceso.** Hace que un determinado proceso se ejecute cada cierto tiempo (segundos, minutos, horas...) por etapas o de una sola vez, pero transcurrido un periodo de tiempo fijo.

- **Despertar un proceso.** Es una forma de desbloquear un proceso que habrá sido bloqueado previamente por temporización o cualquier otra causa.

1.1.5 Prioridades

Todo proceso por sus características e importancia lleva aparejadas unas determinadas necesidades de ejecución en cuanto a urgencia y asignación de recursos.

Las prioridades según los sistemas operativos se pueden clasificar del siguiente modo:

- **Asignadas por el sistema operativo.** Se trata de prioridades que son asignadas a un proceso en el momento de comenzar su ejecución y dependen fundamentalmente de los privilegios de su propietario y del modo de ejecución.
- **Asignadas por el propietario.** En este caso es el propio usuario el que asigna a cada proceso la prioridad con que éste debe ejecutarse. Esta modalidad de asignación de prioridades es muy utilizada en sistema de tiempo real, ya que algunos de sus procesos necesitan atender rápidamente algún evento sin que tengan que interrumpirse.

Otra clasificación de prioridades atendiendo a la posibilidad de variación de las mismas es la siguiente:

- **Estáticas.** No pueden ser modificadas durante la ejecución del proceso. Pueden ser utilizadas en sistemas de tiempo compartido, pero no en los de tiempo real.
- **Dinámicas.** La prioridad de un proceso puede ser modificada con el fin de atender cualquier evento que se produzca.

1.1.6 Clasificación de procesos

En términos generales los procesos se pueden clasificar en dos conjuntos:

- **Procesos limitados por E/S.** Son aquellos procesos que pasan más tiempo realizando E/S que haciendo cálculos. Por ejemplo aplicaciones de bases de datos, aplicaciones comerciales etc.
- **Procesos limitados por CPU.** Son aquellos procesos que pasan el mayor tiempo haciendo cálculos, es decir ocupando CPU. Por ejemplo aplicaciones científicas, de ingeniería etc.

Si todos los procesos son limitados por E/S, la cola ready pasaría vacía y el itinerador de CPU no tendría nada que hacer.
 Si todos los procesos son limitados por CPU, la cola de espera de E/S pasaría vacía y el sistema estaría también desbalanceado.
El mejor desempeño se logra con una buena mezcla de las dos clases de procesos.

1.1.7 Estructuras de datos de procesos: La PCB

Cada proceso tiene asociada una estructura de datos llamada la **PCB** (Process Control Block).

Un proceso se representa desde el punto de vista del sistema operativo, por un conjunto de datos donde se incluyen el estado en cada momento, recursos utilizados, registros, etc., denominado Bloque de Control de Procesos (PCB).

Los objetivos del bloque de control de procesos son los siguientes:

- Localización de la información sobre el proceso por parte del sistema operativo.
- Mantener registrados los datos del proceso en caso de tener que suspender temporalmente su ejecución o reanudarla.

La PCB contiene toda la información que necesita el proceso. Por ejemplo:

- **PID.** Identificador del proceso.
- **Estado.** Wait, Run, Ready etc.
- **PC (Program Counter).** Información relativa al contenido del contador de programa.
- **Registros de CPU.** Archivo de registros en uso.
- **Información de itineración (Información para el Scheduler).** Contadores, relojes, prioridad, punteros a colas de ejecución.
- **Información de Manejo de Memoria.** Registros base y límite, tablas para manejo de memoria virtual, lista de páginas, etc.
- **Información de Contabilidad.** Tiempo de CPU utilizado, tiempo real utilizado, límites de tiempo, etc.
- **Información sobre el estado de E/S.** Dispositivos asignados al proceso, lista de archivos abiertos, estado de esos archivos, etc.
- **Credenciales.** UID, GID, PPID (identificadores de usuario y de proceso).

Estas informaciones se encuentran en memoria principal en disco y se accede a ellas en los momentos en que se hace necesaria su actualización o consulta. Los datos relativos al estado del proceso siempre se encuentran en memoria principal. El concepto del PCB es seguir el **estado de ejecución** y la locación del proceso.

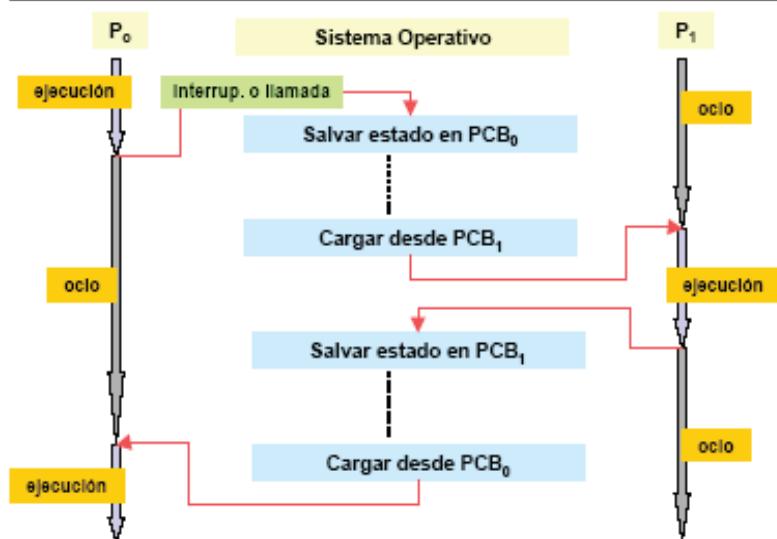
Cuando un proceso se crea, un nuevo PCB se crea para el proceso. Al terminar el proceso, su PCB asociado se destruye. El PCB se coloca en una **cola de estados**.

Existe un **Bloque de Control de Sistema (SCB)** con objetivos similares al anterior y entre los que se encuentra el enlazado de los bloques de control de procesos existentes en el sistema.

El cambio de contexto se producirá en caso de ejecución de una instrucción privilegiada, una llamada al sistema operativo o una interrupción, es decir, siempre que se requiera la atención de algún servicio del sistema operativo.

Gráfica 35. Comutación y la PCB

Comutación de la CPU entre Procesos



1.2 Hebras de control (Threads)

Un proceso está definido por sus recursos y por su ubicación en el espacio de memoria. Un thread es una unidad básica de utilización de CPU.

Los threads comparten código, datos y otros recursos como archivos. El conjunto de threads se denomina tarea (task).

Los procesos normales se denominan procesos pesados (HWP). Los threads se denominan procesos livianos.

Un proceso pesado equivale a una tarea con un solo thread.

Un thread siempre pertenece a sólo una tarea.

Para realizar la misma tarea muchas veces: es más eficiente tener un proceso y varias threads que varios procesos diferentes (por ejemplo, un servidor de red http, ft, etc): es mas rápido hacer los cambios de contexto entre threads.

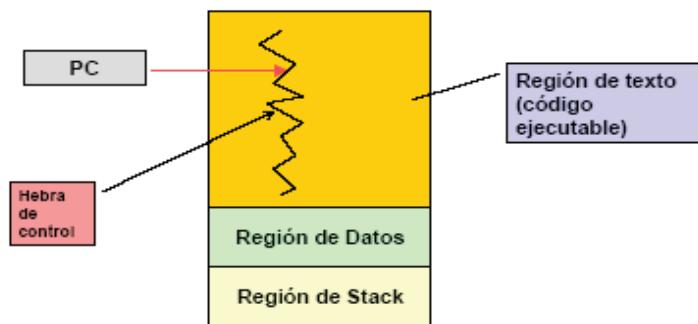
Al compartir el espacio de memoria, es fácil y simple cooperar entre threads (no es necesario que hagan llamadas al sistema para este fin).

Su estructura es similar a la de los procesos, poseen estados (corriendo, lista para correr, esperando, etc) y pueden crear thread hijas, etc. Pero difieren de los procesos ya que no hay protección de memoria, esto es responsabilidad de programador.

Gráfica 36. Estructura de un proceso

El Concepto de Threads

Un proceso normal tiene la siguiente estructura



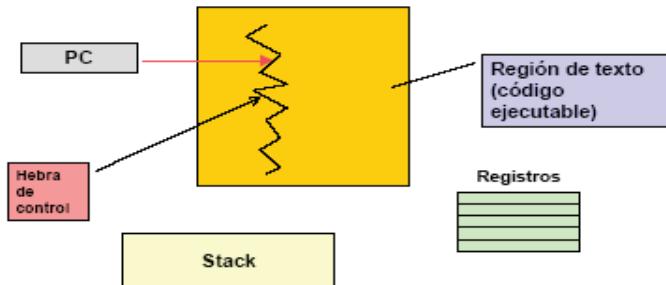
Las threads se encuentran en la región destinada al código ejecutable, y son las que forman un proceso, ya sea una sola o varias, esto es, según la necesidad y la forma en que opera el sistema operativo. Cada una debe tener asignada un PC (contador de programa).

Un thread maneja recursos, que le son útiles para la ejecución de su tarea específica, estos tienen que ver con la memoria y los registros. Veamos:

Gráfica 37. Recursos de un thread

Recursos de un Thread

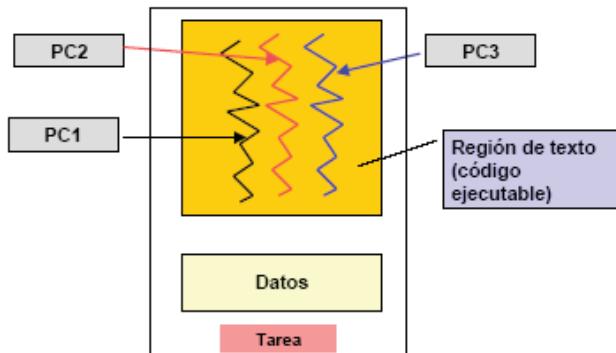
Un Thread tiene los siguientes recursos:



Como se dijo puede ser que una tarea tenga más de un thread, en este caso cada Thread es independiente y sigue manejando sus propios recursos y sus propias prioridades.

Gráfica 37. Tarea con múltiples threads

Tarea con múltiples Threads



1.3 Comunicación entre procesos

Los procesos pueden cooperar en la realización de una tarea. Una forma de cooperación es a través de la facilidad llamada **comunicación entre procesos** (IPC Inter Process Communication).

El IPC provee la forma a través de la cual los procesos pueden comunicarse y sincronizarse.

1.3.1 Mensajes

La comunicación vía IPC no necesita compartir variables. Para esto se requieren dos operaciones como mínimo:

- Send (**mensaje**) – Enviar (mensaje)
- Receive (**mensaje**) – Recibir (mensaje)

Los procesos que se comunican necesitan una forma de poder reverenciarse.

La comunicación puede ser:

1. Comunicación directa

En este esquema de comunicación, cada proceso debe explícitamente indicar el nombre del proceso fuente o del proceso destino:

- Enviar(P,mensaje); : Envía mensaje al proceso P
- Recibir(Q,mensaje); : Recibe un mensaje desde el proceso Q

2. La interacción productor consumidor

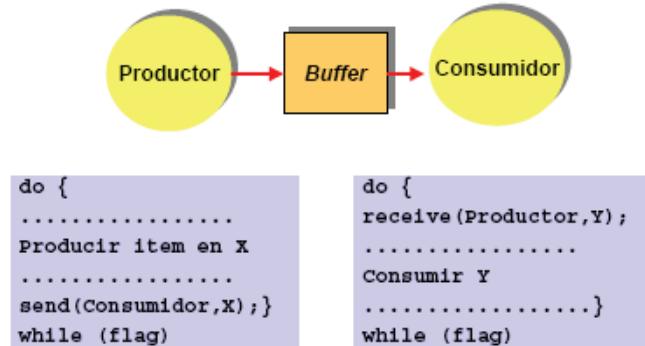
Un tipo de interacción muy común en un sistema operativo es la llamada Productor/Consumidor.

El proceso **productor** genera algún dato y el proceso **consumidor** recibe este dato para algún tipo de procesamiento.

Por ejemplo:

- **Productor**: aplicación que genera datos.
- **Consumidor**: impresora que imprime los datos.

Como los procesos productor y consumidor tienen distintas velocidades, normalmente se necesita un Buffer para suavizar esta diferencia. El IPC provee de este Buffer de forma automática

Gráfica 38. Interacción productor consumidor**La Interacción Productor Consumidor****3. Direccionamiento simétrico y asimétrico**

El esquema visto muestra una simetría: tanto los procesos transmisor como receptor tienen que nombrarse explícitamente. Una variación consiste en que sólo el proceso que envía señala explícitamente el nombre del receptor y el receptor en vez de poner el nombre del transmisor pone una variable.

- Enviar(P, mensaje)
- Recibir(id, mensaje) la variable id registra el nombre del proceso que envía.

Gráfica 39. Direccionamiento asimétrico- Cliente/servidor

La desventaja tanto del método simétrico como asimétrico es la baja modularidad. Si el nombre de un proceso cambia, es necesario cambiar todas las definiciones de nombre internas.

Una forma alternativa es la llamada **comunicación indirecta**.

4. Comunicación indirecta

En el método de comunicación indirecta, los mensajes se envían y reciben desde objetos llamados *mailbox* (o puertas). Cada mailbox tiene una identificación única. Dos procesos sólo se pueden comunicar si ambos comparten un mailbox.

- Enviar(A,mensaje) : se envía un mensaje al mailbox A
- Recibir(A,mensaje) : se recibe un mensaje desde el mailbox A

1.3.2 Aspectos de comunicación

El sistema *IPC* es un sistema de comunicación, es decir se requiere de un enlace de comunicación entre dos procesos.

Un enlace de comunicación puede ser considerado como una cola de mensajes.

Existen varias formas de implementar esta cola de mensajes:

- Capacidad cero: sincronización *rendezvous*
- Capacidad limitada
- Capacidad ilimitada

Capacidad cero: sincronización *rendezvous*: Los procesos que envían y reciben quedan bloqueados.

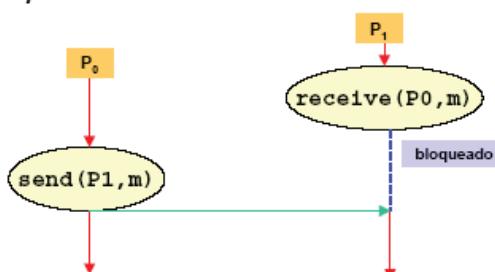
Capacidad limitada y Capacidad ilimitada: es necesario señalar cuando un mensaje ha llegado a su destino. Esto se puede hacer vía un mensaje corto llamado *ACK*. O también llamada sincronización con buffers.

Gráfica 40. Sincronización Rendezvous

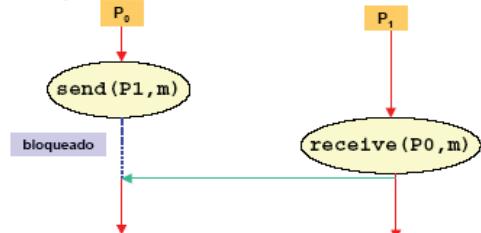
Sincronización Rendezvous

Sincronización Rendezvous

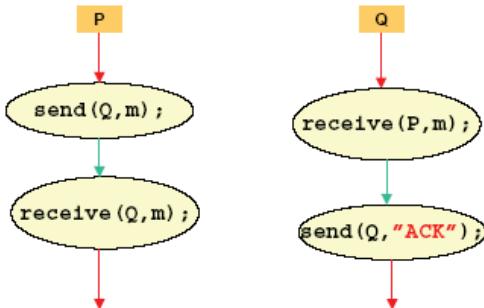
Si P_1 ejecuta *receive* antes de *send*, queda bloqueado



Si P_0 ejecuta *send* antes de *receive*, queda bloqueado



Gráfica 41. Sincronización con buffers
Sincronización con Buffers



1.4 Sincronización de procesos

Se denominan **procesos cooperativos** a procesos que cooperan en torno a una tarea común. Procesos cooperativos necesitan compartir datos, y lo pueden hacer a través de compartir espacio de memoria o compartir archivos.

El acceso concurrente a datos compartidos puede generar inconsistencias de datos, provocando errores severos y difíciles de detectar.

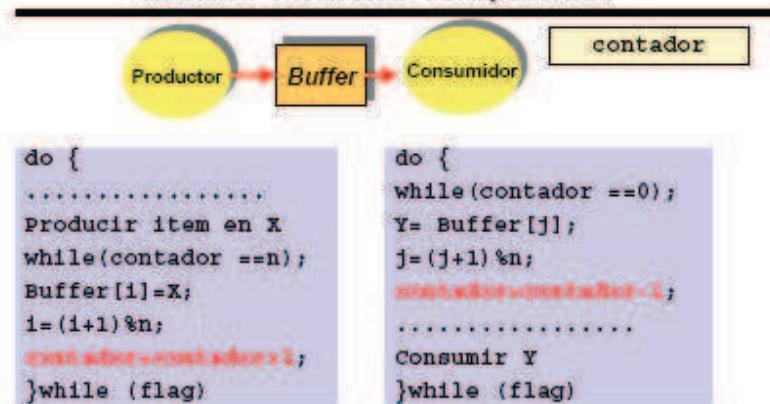
Un ejemplo de procesos cooperativos es la relación productor/consumidor.

Analizaremos una solución usando una variable compartida.

Es necesario recordar que ambos procesos se realizan de forma asincrónica.

Gráfica 42. Ejemplo de variables compartidas

**La Interacción Productor Consumidor
usando variables compartidas**



Si se analiza la solución anterior, ambas rutinas funcionan bien, pero si se ejecutan en forma concurrente se pueden producir anomalías.

La variable compartida es **contador**. Supongamos que el valor es 5. El productor ejecuta `contador=contador+1` y el consumidor ejecuta `contador=contador-1` concurrentemente. ¿**Qué ocurre?**

La ejecución concurrente de: `contador=contador+1` y `contador=contador-1` es equivalente a la ejecución secuencial donde se mezclan instrucciones de ambos procesos.

De esta forma, con cada uno de los procesos anteriores la variable contador quedaría con valores diferentes (asumiendo que su valor inicial sea 5), en el primer proceso quedaría con valor 6 y en el segundo con valor 4. Como se ejecutan al tiempo, se genera un error.

1.4.1 La sección crítica

En el ejemplo anterior, ambos procesos tienen una parte de su código en la cual acceden variables compartidas. Esta parte del código se denomina **Sección Crítica**.

Las anomalías se producen cuando dos o más procesos ejecutan en forma concurrente su **Sección Crítica**.

Gráfica 43. Ejemplo de sección crítica

Ejemplo de Sección Crítica

El acceso a **contador** es la Sección Crítica del problema

```
do {
    .....
    Producir item en X
    while(contador ==n);
    Buffer[i]=X;
    i=(i+1)%n;
    contador=contador+1;
}while (flag)
```

```
do {
    while(contador ==0);
    Y= Buffer[j];
    j=(j+1)%n;
    contador=contador-1;
    .....
    Consumir Y
}while (flag)
```

Solución al problema de la sección crítica

Para evitar anomalías en la ejecución de las secciones críticas de los procesos, cuando un proceso está en su sección crítica, ningún otro proceso puede ejecutar su sección crítica.

La ejecución de las secciones críticas, debe ser **mutuamente exclusiva** en el tiempo. Si dos o más procesos desean cooperar, es necesario disponer de un protocolo.

Requisitos de solución

Una solución al problema de la sección crítica debe satisfacer los siguientes tres requisitos:

1. **Exclusión mutua.** Si un proceso P_i está en su sección crítica, ningún otro proceso puede estar ejecutando su sección crítica.
2. **Progreso.** Si un proceso P_i está en su sección crítica, y otro proceso desea entrar a su sección crítica, sólo los procesos que no están en el resto del código, pueden participar de la decisión de qué proceso entrará próximamente, y esta selección no se puede postergar indefinidamente.
3. **Espera acotada.** Un proceso no puede quedar esperando indefinidamente a entrar en su sección crítica.

1.4.2 Semáforos

Las soluciones a los problemas de secciones críticas son difíciles de generalizar a problemas de mayor complejidad.

Herramientas de sincronización mucho más flexibles son los llamados semáforos.

Un **semáforo** es una variable entera que es accesada a través de dos operaciones atómicas llamadas **wait** y **signal**.

Las operaciones wait y signal

```
wait(S) : while (S<=0);
```

```
    S=S-1;
```

```
signal(S) : S=S+1;
```

Las modificaciones a la variable entera en wait y signal se ejecutan en forma indivisible.

Gráfica 44. Implementación del semáforo

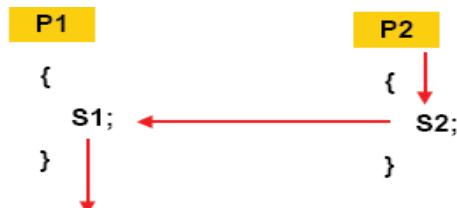
Ejemplo1: n secciones críticas

Los n-procesos comparten el semáforo llamado mutex. Este semáforo se inicializa en el valor 1

```
do {
    wait(mutex);
    Sección crítica
    signal(mutex);
    Resto del código
}while (flag)
```

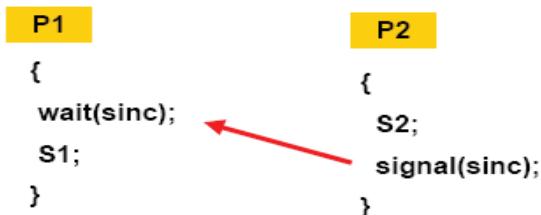
Un problema es la exclusión mutua de secciones críticas. Otro problema diferente y frecuente es la **sincronización de procesos**. Sincronizar procesos significa forzar un orden en su ejecución.

Gráfica 45. Implementación de semáforo para la sincronización



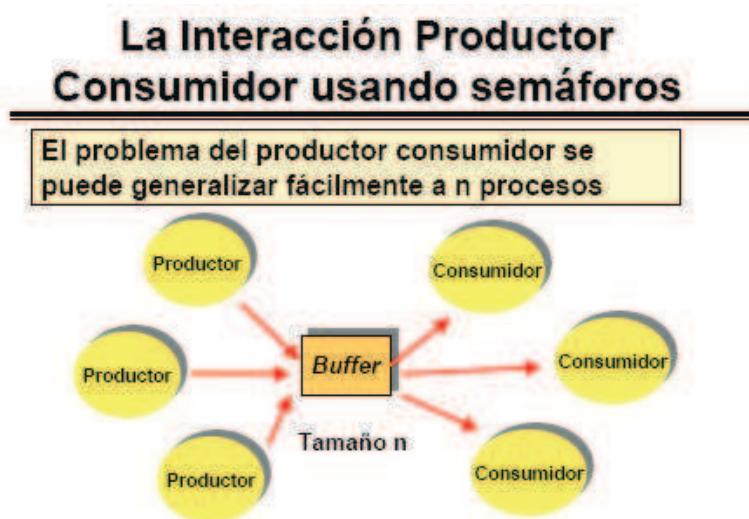
Queremos obligar a que S1 se ejecute sólo después que S2 se haya ejecutado. ¿Cómo se puede hacer?

```
semaforo sinc=0;
```



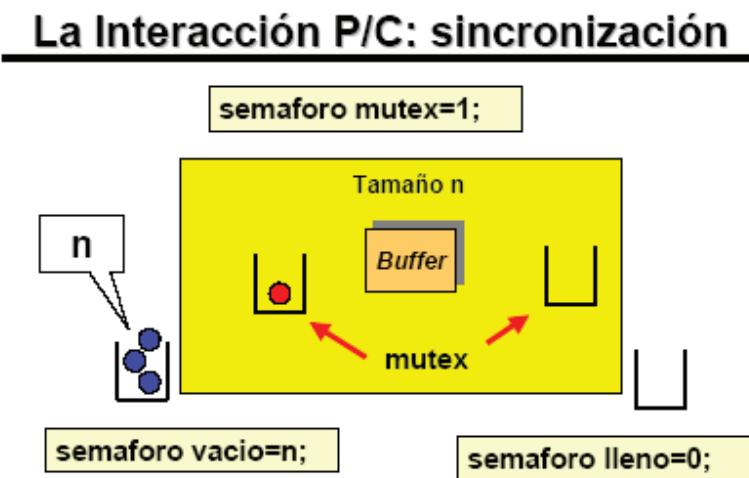
En la interacción productor/consumidor con semáforos hay que proporcionar exclusión mutua al acceso al buffer.

Gráfica 46. Implementación de semáforo en productor/consumidor



Este acceso se logra con un semáforo binario inicializado en 1. Además del acceso al buffer hay que sincronizar para evitar que un proceso escriba en el buffer si éste está lleno o que un proceso lea el buffer si está vacío.

Gráfica 47. Productor/consumidor: sincronización completa



1.5 Planificación de procesos – Scheduling – Itineración de procesos

La planificación del procesador o de CPU se refiere a la manera o técnicas que se usan para decidir cuánto tiempo de ejecución y cuando se le asignan a cada proceso del sistema. Obviamente, si el sistema es monousuario y monotarea no hay mucho que decidir, pero en el resto de los sistemas esto es crucial para el buen funcionamiento del sistema.

1.5.1 Niveles de planificación

En los sistemas de planificación generalmente se identifican tres niveles: el alto, el medio y el bajo.

El **nivel alto** decide que trabajos (conjunto de procesos) son candidatos a convertirse en procesos compitiendo por los recursos del sistema.

El **nivel intermedio** decide que procesos se suspenden o reanudan para lograr ciertas metas de rendimiento.

El planificador de **bajo nivel** es el que decide qué proceso, de los que ya están listos (y que en algún momento pasó por los otros dos planificadores) es al que le toca ahora estar ejecutándose en la unidad central de procesamiento.

1.5.2 Objetivos de la planificación

Una estrategia de planificación debe buscar que los procesos obtengan sus turnos de ejecución apropiadamente, conjuntamente con un buen rendimiento y minimización de la sobrecarga (overhead) del planificador mismo. En general, se buscan cinco objetivos principales:

- **Justicia o imparcialidad.** Todos los procesos son tratados de la misma forma, y en algún momento obtienen su turno de ejecución o intervalos de tiempo de ejecución hasta su terminación exitosa.
- **Maximizar la producción.** El sistema debe de finalizar el mayor numero de procesos en por unidad de tiempo.
- **Maximizar el tiempo de respuesta.** Cada usuario o proceso debe observar que el sistema les responde consistentemente a sus requerimientos.
- **Evitar el aplazamiento indefinido.** Los procesos deben terminar en un plazo finito de tiempo.
- **El sistema debe ser predecible.** Ante cargas de trabajo ligeras el sistema debe responder rápido y con cargas pesadas debe ir degradándose

paulatinamente. Otro punto de vista de esto es que si se ejecuta el mismo proceso en cargas similares de todo el sistema, la respuesta en todos los casos debe ser similar.

1.5.3 Características a considerar de los procesos

No todos los equipos de cómputo procesan el mismo tipo de trabajos, y un algoritmo de planificación que en un sistema funciona excelente puede dar un rendimiento pésimo en otro cuyos procesos tienen características diferentes. Estas características pueden ser:

- **Cantidad de Entrada/Salida.** Existen procesos que realizan una gran cantidad de operaciones de entrada y salida (aplicaciones de bases de datos, por ejemplo).
- **Cantidad de uso de CPU.** Existen procesos que no realizan muchas operaciones de entrada y salida, sino que usan intensivamente la unidad central de procesamiento. Por ejemplo, operaciones con matrices.
- **Procesos de lote o interactivos.** Un proceso de lote es más eficiente en cuanto a la lectura de datos, ya que generalmente lo hace de archivos, mientras que un programa interactivo espera mucho tiempo (no es lo mismo el tiempo de lectura de un archivo que la velocidad en que una persona teclea datos) por las respuestas de los usuarios.
- **Procesos en tiempo real.** Si los procesos deben dar respuesta en tiempo real se requiere que tengan prioridad para los turnos de ejecución.
- **Longevidad de los procesos.** Existen procesos que típicamente requieren varias horas para finalizar su labor, mientras que existen otros que sólo necesitan algunos segundos.

1.5.4 Planificación apropiativa o no apropiativa (preemptive or not preemptive)

La planificación **apropiativa** es aquella en la cual, una vez que a un proceso le toca su turno de ejecución ya no puede ser suspendido, ya no se le puede arrebatar la unidad central de procesamiento. Este esquema puede ser peligroso, ya que si el proceso contiene accidental o deliberadamente ciclos infinitos, el resto de los procesos pueden quedar aplazados indefinidamente.

Una planificación **no apropiativa** es aquella en que existe un reloj que lanza interrupciones periódicas en las cuales el planificador toma el control y se decide si el mismo proceso seguirá ejecutándose o se le da su turno a otro proceso. Este mismo reloj puede servir para lanzar procesos manejados por el reloj del sistema.

Por ejemplo en los sistemas UNIX existen los “cronjobs” y “atjobs”, los cuales se programan con base a la hora, minuto, día del mes, día de la semana y día del año.

La mayoría de los sistemas programan el reloj para que pulse de 20 a 50 veces por segundo y al pulso final es cuando el planificador consigue el control y puede relegar al proceso en ejecución.

Las prioridades de los procesos se recalcularn frecuentemente. En el cálculo de prioridades se tiene en cuenta la duración del lapso de tiempo. No tendría sentido asignar continuamente la CPU a un proceso y luego expulsar el proceso cuando tan sólo haya ejecutado unas pocas instrucciones.

En una planificación **no apropiativa**, un trabajo muy grande aplaza mucho a uno pequeño, y si entra un proceso de alta prioridad esté también debe esperar a que termine el proceso actual en ejecución.

1.5.5 Planeadores – Scheduler - Itineradores

El objetivo de la **multiprogramación** es maximizar la utilización de la CPU manteniéndola ocupada la mayor parte del tiempo.

El objetivo de un sistema de **tiempo compartido** es conmutar la CPU en forma rápida para que cada usuario pueda interactuar con su aplicación.

Cuando un proceso entra al sistema, es puesto en una cola de jobs. Los procesos que residen en memoria y están listos para ser ejecutados están en una cola llamada **Cola Ready**.

Los procesos que están a la espera por dispositivo de E/S, están en una cola llamada **Cola de Dispositivos**.

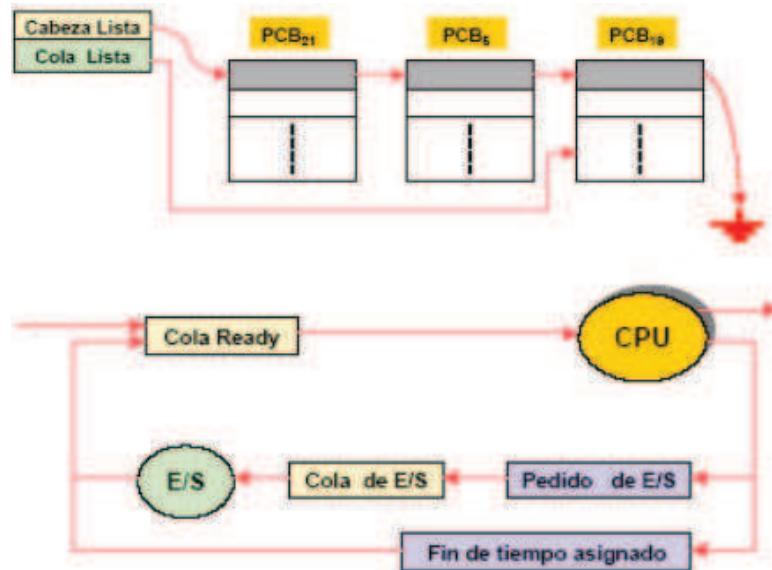
De la cola ready los procesos son seleccionados para su ejecución y despachados a la CPU. El ciclo de vida se puede observar mediante un diagrama de colas.

Un proceso durante su ciclo de vida en el sistema pasa por varias colas. El S.O debe seleccionar procesos desde estas colas con algún criterio.

La selección de un proceso la realiza el **itinerador (scheduler)**.

Existen dos itineradores:

- **De Job.** Selecciona un proceso del disco y lo carga en memoria.
- **De CPU.** Selecciona un proceso de la cola ready y le asigna CPU.

Gráfica 48. Estructura de una cola

El itinerador de CPU

El Itinerador de CPU actúa frecuentemente. Normalmente un proceso sólo ocupa la CPU algunos milisegundos antes de esperar por E/S.

El itinerador de CPU debe ser muy eficiente. Si normalmente un proceso toma 100mseg antes de requerir E/S, el tiempo que debe tomar el itinerador no debe superar el 10% de este tiempo, es decir 10mseg

El itinerador de JOB

El Itinerador de CPU actúa menos frecuentemente que el itinerador de CPU. El itinerador de Job controla el grado de multiprogramación: número de jobs presentes en memoria.

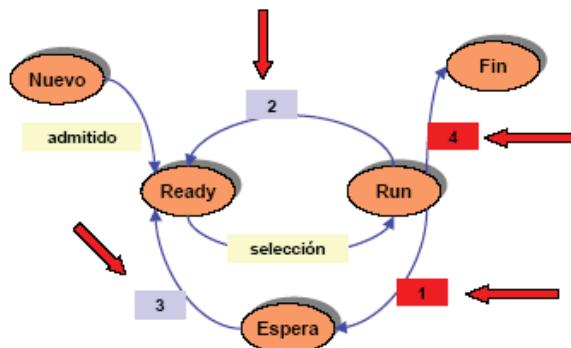
Si el grado de multiprogramación es estable, la tasa de creación de procesos es igual a la tasa de procesos que terminan.

Un aspecto importante del itinerador de Job es el criterio de selección de jobs.

1.5.6 Cuándo se itinera - La cola ready

Toda vez que la CPU esté ociosa, el itinerador selecciona un proceso desde la *Cola Ready*. La *Cola Ready* no es necesariamente una cola FIFO. Su estructura varía dependiendo del algoritmo de itineración.

**Gráfica 49. Cuándo se itinerá
¿Cuándo se itinerá?**



En 1 y 4 no hay alternativa. Si la Cola Ready no está vacía, se debe seleccionar un proceso. En estas condiciones, se dice que la itineración es **no interruptible**.

No interruptible significa una vez que la CPU se asignó a un proceso, lo abandona ya sea porque terminó, o porque se pasó a estado de espera.

En 2 y 3 es posible tomar decisiones. Se dice que la itineración es **interruptible**. La itineración **interruptible** es más difícil de implementar porque se debe mantener la consistencia en las estructuras de datos. Por ejemplo el kernel del S.O

1.5.7 Módulo de despacho (Dispatcher)

El despachador es el módulo del S.O que entrega el control de la CPU a los procesos seleccionados.

Este módulo tiene un impacto importante en el desempeño, ya que es invocado en el *context switch*.

El tiempo que toma parar un proceso y comenzar la ejecución de otro se conoce como **latencia de despacho**.

1.5.8 El context switch

Un Intercambio de contexto consiste en **cambiar el proceso** que se está ejecutando. Involucra guardar el estado del proceso que se estaba ejecutando y restaurar el estado del proceso que se va a ejecutar.

El tiempo que tarda este proceso no es despreciable, y es tiempo perdido. Se realiza con soporte del hardware. Los tiempos típicos: 1 a 1000E-6 s. Un esfuerzo importante en el diseño de un sistema operativo es minimizar el tiempo de context switch.

Cuanto más complejo el sistema (de hardware), más operaciones debe realizar el cambio de contexto (guardar/restaurar mas registros en memoria).

1.5.9 Criterios de itineración - Planificación

- ¿Qué algoritmo seleccionar?
- ¿Cómo comparar distintos algoritmos de itineración?

Para cuantificar se necesita tener criterios claros. Los criterios son:

- **Utilización de la CPU.** Porcentaje del tiempo que la CPU está ocupada
- **Throughput.** Número de procesos por unidad de tiempo.
- **Tiempo turnaround.** Intervalo de tiempo desde la entrega del proceso al sistema hasta su finalización.
- **Tiempo de espera:** Suma de todos los tiempos que un proceso espera en la cola sin ocupar CPU.
- **Tiempo de respuesta.** Es el tiempo que transcurre entre un requerimiento hasta el comienzo de la respuesta.

1.5.10 Algoritmos de itineración - Planificación

¿Cuál de los procesos que están en la **cola ready** se escogerá para asignarle CPU?

Existen varios algoritmos. Estos son:

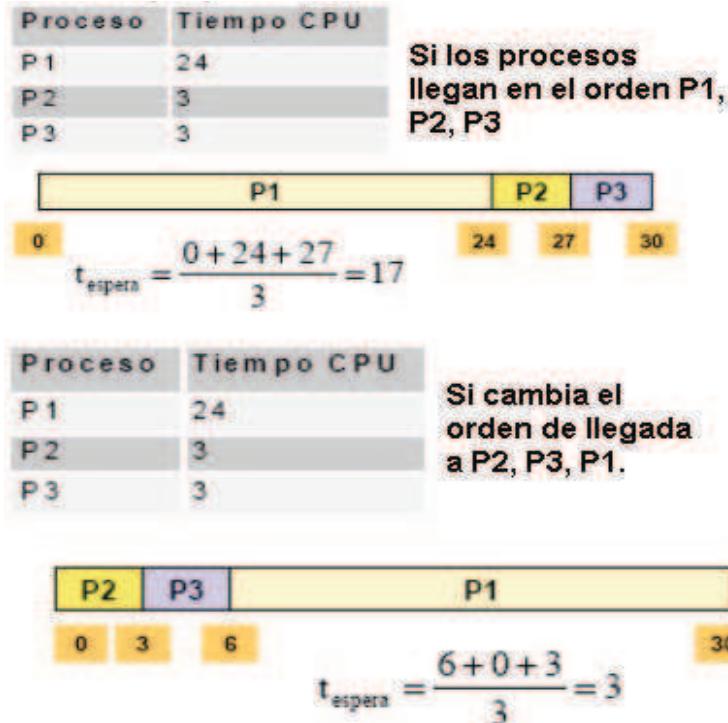
- Peluquería
- El más corto primero
- Prioridad
- Round Robin
- Colas multinivel

1. Algoritmo de la peluquería – Primero en llegar primero en ser servido

También se llama FCFS (el primero que llega es el primero en ser atendido). La implementación es muy simple porque se maneja con una cola del tipo FIFO.

Si bien es simple el tiempo de espera promedio es elevado

Gráfica 50. Orden de peluquería



El tiempo de espera de este algoritmo no es mínimo.

En un escenario dinámico, si un proceso ocupa más tiempo que los demás, arrastra a los demás produciendo un efecto llamado *convoy*.

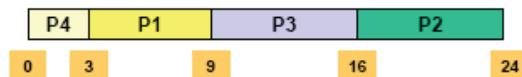
La itineración orden de peluquería es no interrumpible.

2. Algoritmo El Más Corto Primero

Este algoritmo asocia con cada proceso el tiempo de CPU que ocupará la próxima vez. Si más de un proceso tiene el mismo tiempo, se rompe el empate usando orden de peluquería en la itineración.

Gráfica 51. El más corto primero**Ejemplo**

Proceso	Tiempo CPU	El tiempo está en mseg
P1	6	
P2	8	
P3	7	
P4	3	



$$t_{\text{espera}} = \frac{3+16+9+0}{4} = 7$$

El algoritmo el más corto primero es óptimo, en el sentido que entrega el tiempo de espera mínimo para un conjunto de procesos.

La dificultad está en saber cuál será el tiempo que ocupará el proceso en su próxima utilización de CPU.

Normalmente se utiliza este algoritmo en la itineración de job (Disco → Memoria)

3. Itineración por Prioridad

El algoritmo el más corto primero es un caso especial de la itineración por prioridad.

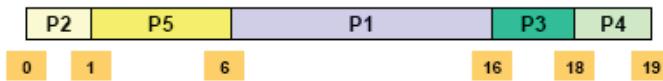
Se asocia una prioridad a cada proceso, y se le asigna la CPU al proceso que tiene mayor prioridad.

Los procesos con la misma prioridad se itineran según orden de peluquería.

La prioridad se expresa con un rango de números. Generalmente los números menores representan alta prioridad.

Gráfica 52. Por prioridad

Proceso	Tiempo CPU	Prioridad
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2



$$t_{\text{espera}} = \frac{6+0+16+18+1}{5} = \frac{41}{5} = 8.2 \text{ mseg}$$

Las prioridades se pueden definir en forma interna o externa.
Internamente implica definir criterios como por ejemplo: límites de tiempo, requerimientos de memoria etc.
Externamente se fijan criterios como: importancia del proceso, grupo de trabajo etc.
La itineración por prioridad puede ser interrumpible o no interrumpible
Con un algoritmo interrumpible, la CPU se transfiere a otro proceso, si la prioridad del proceso que llega, es mayor que la actual.
Un algoritmo no interrumpible significa poner el nuevo proceso en la cabeza de la cola ready.
Un problema con el algoritmo de prioridad es que puede producir un *bloqueo indefinido*.
Para solucionar este problema se usa la técnica de aumentar gradualmente la prioridad de los procesos que esperan.

4. Itineración Round Robin

Esta itineración se diseña especialmente para sistemas de tiempo compartido. Es similar al orden de peluquería, pero se interrumpe la CPU para conmutar entre los procesos.

Se define una unidad de tiempo llamada quantum tq

$$10 \text{ mseg} \leq t_q \leq 100 \text{ mseg}$$

La cola ready es tratada como una cola circular. El itinerador recorre la cola asignando CPU a cada proceso un intervalo de tiempo dado por el cuantum de tiempo.

La Cola Ready es tratada como cola FIFO. Los procesos que entran se agregan al final. Si el tiempo que ocupa un proceso es menor que el cuantum, el proceso abandona la CPU voluntariamente y el itinerador toma el siguiente proceso.

El Algoritmo Round Robin (RR) es interrumpible.
 Si hay n procesos en la cola, cada proceso obtiene $1/n$ del tiempo de CPU en cada ciclo de a lo más q unidades de tiempo.
 Cada proceso no espera más que $(n - 1)*q$ hasta el siguiente cuantum.
 El desempeño depende del tamaño de t_q .
 Si t_q es muy grande, infinito, RR se transforma en itineración por orden de peluquería.
 Si t_q es muy chico, RR se transforma en un algoritmo que comparte la CPU, de forma tal que si hay n procesos, cada usuario percibe que la velocidad es $1/n$ de la velocidad del procesador.
 Es necesario considerar el tiempo que demora el *context switch*.
 Si el tiempo de context switch es el 10% de t_q , aproximadamente el 10% del tiempo de CPU se gasta en context switch.

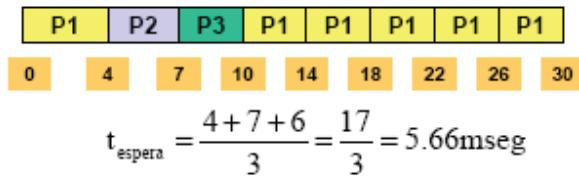
Gráfica 53. Round Robin

Evaluación

El tiempo de espera promedio es en general alto. Ejemplo:

Proceso	Tiempo CPU
P1	24
P2	3
P3	3

Los procesos P1, P2, y P3 llegan al mismo tiempo.
 El cuantum es $t_q = 4$ mseg



$$t_{\text{espera}} = \frac{4+7+6}{3} = \frac{17}{3} = 5.66 \text{mseg}$$

5. Algoritmo Colas *Multinivel*

Una clase distinta de algoritmo se necesita cuando los procesos son clasificados en diferentes grupos.

En un sistema normalmente coexisten dos tipos de procesos:

- Interactivos
- Batch

Ambos tipos de procesos requieren distintos tiempos de respuesta, y por lo tanto sus necesidades de itineración son distintas.

El Algoritmo de colas multinivel, partitiona la Cola Ready en múltiples colas.

Cada proceso es asignado permanentemente a una cola, basándose en alguna propiedad como por ejemplo tamaño, tipo o prioridad.

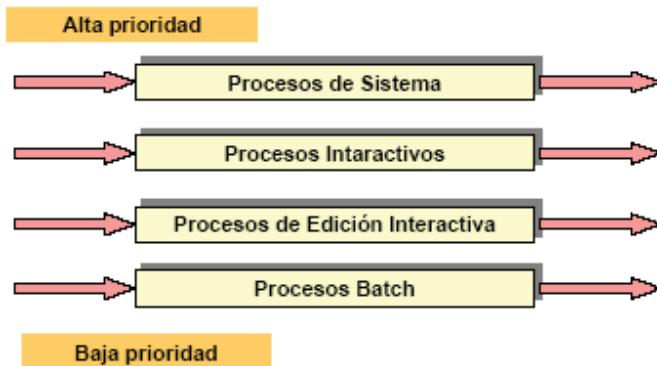
Cada Cola tiene un algoritmo distinto.

Por ejemplo procesos Batch se pueden itinerar por orden de peluquería mientras que los procesos interactivos se pueden itinerar usando RR.

Los procesos interactivos tienen prioridad sobre los procesos Batch.

Gráfica 54. Colas multinivel

Algoritmo Colas Multinivel: Ejemplo

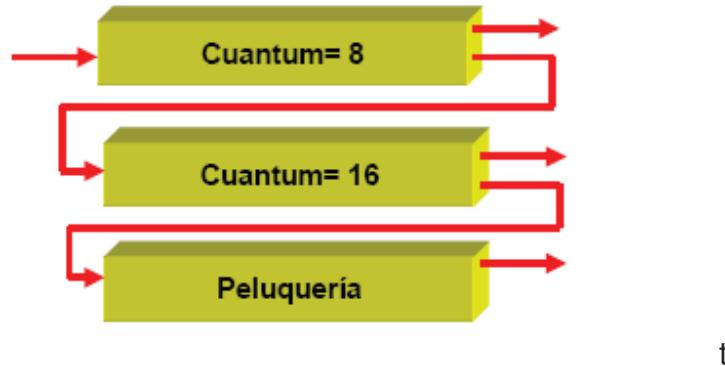


Algoritmo Colas Multinivel Realimentadas

En el algoritmo de colas multinivel, los procesos son asignados permanentemente en el momento que entran al sistema.

El algoritmo de colas multinivel realimentadas permite que los procesos se muevan entre las colas.

La idea es separar procesos con distintos tiempos de CPU.

Gráfica 55. Colas multinivel realimentadas**Algoritmo Colas Multinivel Realimentadas**

El algoritmo de múltiples colas realimentadas está definido por los siguientes parámetros:

- Número de colas.
- El algoritmo de itineración de cada cola.
- Métodos para cambiar de colas a procesos.
- Método para determinar a cuál cola entra el proceso.

Este algoritmo es el más general. Se puede configurar para múltiples requerimientos.

Actividad final:

De la lista de procesos realizada inicialmente revise cuáles eran procesos verdaderos y cuáles no. Recuerde que aquí ya debe manejar bien este concepto. Analice el por qué acertó o falló en cada punto, según el caso.
Guarde la lista para la actividad final.

CAPÍTULO 2. ADMINISTRACIÓN DE LA MEMORIA

Actividad inicial:

Sabe a qué hacen referencia los términos: paginación, segmentación,

memoria virtual, reemplazo de páginas?

Si es así coloque en una lista la definición que sepa y al final concluimos.

En este capítulo se describirán las técnicas más usuales en el manejo de memoria, revisando los conceptos relevantes. Se abarcarán los esquemas de manejo simple de memoria real, la multiprogramación en memoria real con sus variantes, el concepto de “**overlays**”, la multiprogramación con intercambio y los esquemas de manejo de memoria virtual.

2.1 Estructura general

La memoria es un importante recurso que debe administrarse con cuidado. Los programas crecen en tamaño tan rápido como las memorias.

La parte del sistema operativo que administra la memoria se llama administrador de la memoria. Su labor consiste en llevar el registro de las partes de memoria que se estén utilizando y aquellas que no, con el fin de asignar espacio en memoria a los procesos; cuando estos la necesiten y liberarlo cuando terminen. Así como administrar el intercambio entre la memoria principal y el disco, en los casos en que la memoria principal no pueda albergar a todos los procesos.

Un vistazo al material que se va a cubrir en este capítulo se muestra en la siguiente gráfica en donde se especifican, en términos generales, los conceptos más importantes en cuanto a las técnicas empleadas en el manejo de memoria.

Gráfica 56. Panorama del manejo de memoria

Real	Real	Virtual	
Mono usuario	Multiprogramación	Multiprogramación	
	Particiones	Paginación Pura	Segmentación Pura
	Fijas Variables	Combinación	
Relocalización		Protección	

2.2 Manejo de memoria en sistemas monousuario sin intercambio

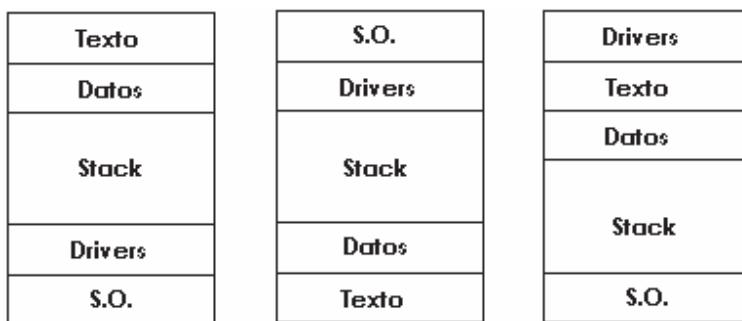
Este esquema se usa principalmente en sistemas monousuario y monotarea, como son las computadoras personales con DOS.

Bajo este esquema, la memoria real es tomada para almacenar el programa que se esté ejecutando en un momento dado, con la visible desventaja de que se está limitado a la cantidad de RAM disponible únicamente.

La organización física bajo este esquema es muy simple: El sistema operativo se ubica en las localidades superiores o inferiores de la memoria, seguido por algunos manejadores de dispositivos (drivers). Esto deja un espacio contiguo de memoria disponible que es tomado por los programas del usuario, dejando generalmente la ubicación de la pila (stack) al último, con el objetivo de que ésta pueda crecer hasta el máximo posible.

Estas diferentes opciones se pueden ver en la siguiente gráfica. Como es obvio, bajo estos esquemas no se requieren algoritmos sofisticados para asignar la memoria a los diferentes procesos, ya que éstos son ejecutados secuencialmente conforme van terminando.

Gráfica 57. Organización simple de memoria



2.3 Multiprogramación en memoria real

En los 60's, las empresas e instituciones que habían invertido grandes sumas en la compra de equipo de cómputo se dieron cuenta rápidamente que los sistemas en lote invertían una gran cantidad de tiempo en operaciones de entrada y salida, donde la intervención de la unidad central de procesamiento era prácticamente nula, y se comenzaron a preguntar cómo hacer que se mantuviera más tiempo ocupada. Fue así como nació el concepto de **multiprogramación**, el cual consiste en la idea de poner en la memoria física más de un proceso al mismo tiempo, de manera que si el que se está ejecutando en este momento entraba en un periodo de entrada/salida, se podía tomar otro proceso para que usara la unidad central de procesamiento. De esta forma, la memoria física se dividía en secciones de tamaño suficiente para contener a varios programas.

De esta manera, si un sistema gastaba en promedio 60% de su tiempo en entrada/salida por proceso, se podía aprovechar más el CPU. Anterior a esto, el

CPU mantenía ese mismo porcentaje ocioso; con la nueva técnica, el tiempo promedio ocioso disminuye de la siguiente forma:

Llámese al tiempo promedio que el CPU está ocupado “grado de multiprogramación”. Si el sistema tuviese un sólo proceso siempre, y éste gastara 60% en entrada/salida, el grado de multiprogramación sería $1 - 60\% = 40\% = 0.4$.

Con dos procesos, para que el CPU esté ocioso se necesita que ambos procesos necesiten estar haciendo entrada/salida, es decir, suponiendo que son independientes, la probabilidad de que ambos estén en entrada/salida es el producto de sus probabilidades, es decir, $0.6 \times 0.6 = 0.36$. Ahora, el grado de multiprogramación es $1 - (\text{probabilidad de que ambos procesos estén haciendo entrada/salida}) = 1 - 0.36 = 0.64$.

Como se ve, el sistema mejora su uso de CPU en un 24% al aumentar de uno a dos procesos. Para tres procesos el grado de multiprogramación es $1 - (0.6)^3 = 0.784$, es decir, el sistema está ocupado el 78.4% del tiempo. La fórmula del grado de multiprogramación, aunque es muy idealista, pudo servir de guía para planear un posible crecimiento con la compra de memoria real, es decir, para obtener el punto en que la adición de procesos a RAM ya no incrementa el uso de CPU.

Dentro del esquema de multiprogramación en memoria real surgieron dos problemas interesantes: la protección y la relocalización.

2.3.1 El problema de la relocalización

Este problema no es exclusivo de la multiprogramación en memoria real, sino que se presentó aquí pero se sigue presentando en los esquemas de memoria virtual también. Este problema consiste en que los programas que necesitan cargarse a memoria real ya están compilados y ligados, de manera que internamente contienen una serie de referencias a direcciones de instrucciones, rutinas y procedimientos que ya no son válidas en el espacio de direcciones de memoria real de la sección en la que se carga el programa.

Esto es, cuando se compiló el programa se definieron o resolvieron las direcciones de memoria de acuerdo a la sección de ese momento, pero si el programa se carga otro día en una sección diferente, las direcciones reales ya no coinciden. En este caso, el manejador de memoria puede solucionar el problema de dos maneras: de manera “estática” o de manera “dinámica”. La solución “estática” consiste en que todas las direcciones del programa se vuelvan a recalcular al momento en que el programa se carga a memoria, esto es, prácticamente se vuelve a recomilar el programa. La solución “dinámica” consiste en tener un registro que guarde la dirección base de la sección que va a contener al programa.

Cada vez que el programa haga una referencia a una dirección de memoria, se le suma el registro base para encontrar la dirección real. Por ejemplo, suponga que el programa es cargado en una sección que comienza en la dirección 100. El programa hará referencias a las direcciones 50, 52, 54. Pero el contenido de esas direcciones no es el deseado, sino las direcciones 150, 152 y 154, ya que ahí comienza el programa. La suma de 100 + 50,..., etc. se hacen al tiempo de ejecución.

La primera solución vale más la pena que la segunda si el programa contiene ciclos y es largo, ya que consumirá menos tiempo en la resolución inicial que la segunda solución en las resoluciones en línea.

2.3.2 El problema de la protección

Este problema se refiere a que, una vez que un programa ha sido cargado a memoria en algún segmento en particular, nada le impide al programador que intente direccionar (por error o deliberadamente) localidades de memoria menores que el límite inferior de su programa o superiores a la dirección mayor; es decir, quiere referenciar localidades fuera de su espacio de direcciones. Obviamente, este es un problema de protección, ya que no es legal leer o escribir en áreas de otros programas.

La solución a este problema también puede ser el uso de un registro base y un registro límite. El registro base contiene la dirección del comienzo de la sección que contiene al programa, mientras que el límite contiene la dirección donde termina. Cada vez que el programa hace una referencia a memoria se checa si cae en el rango de los registros y si no es así se envía un mensaje de error y se aborta el programa.

2.3.3 Particiones fijas o particiones variables

En el esquema de la multiprogramación en memoria real se manejan dos alternativas para asignarle a cada programa su partición correspondiente: **particiones de tamaño fijo o particiones de tamaño variable**.

La alternativa más simple son las **particiones fijas**. Dichas particiones se crean cuando se enciende el equipo y permanecerán con los tamaños iniciales hasta que el equipo se apague. Es una alternativa muy vieja, quien hacía la división de particiones era el operador analizando los tamaños estimados de los trabajos de todo el día. Por ejemplo, si el sistema tenía 512 Kb de RAM, podía asignar 64 Kb para el sistema operativo, una partición más de 64 Kb, otra de 128 Kb y una mayor de 256 Kb.

Esto era muy simple, pero inflexible, ya que si surgían trabajos urgentes, por ejemplo, de 400Kb, tenían que esperar a otro día o reparticionar, inicializando el equipo desde cero.

La otra alternativa, que surgió después y como necesidad de mejorar la alternativa anterior, era crear particiones contiguas de **tamaño variable**.

Para esto, el sistema tenía que mantener ya una estructura de datos suficiente para saber en dónde habían huecos disponibles de RAM y de dónde a dónde habían particiones ocupadas por programas en ejecución. Así, cuando un programa requería ser cargado a RAM, el sistema analizaba los huecos para saber si había alguno de tamaño suficiente para el programa que quería entrar, si era así, le asignaba el espacio. Si no, intentaba relocalizar los programas existentes con el propósito de hacer contiguo todo el espacio ocupado, así como todo el espacio libre y así obtener un hueco de tamaño suficiente. Si aún así el programa no cabía, entonces lo bloqueaba y tomaba otro.

El proceso con el cual se juntan los huecos o los espacios ocupados se le llama "**compactación**".

Existen varios algoritmos para elegir el mejor tamaño de partición para un programa, en memoria real y son.

- **Primer ajuste:** Se asigna el primer hueco que sea mayor al tamaño deseado.
- **Mejor ajuste:** Se asigna el hueco cuyo tamaño excede en la menor cantidad al tamaño deseado. Requiere de una búsqueda exhaustiva.
- **Peor ajuste:** Se asigna el hueco cuyo tamaño excede en la mayor cantidad al tamaño deseado. Requiere también de una búsqueda exhaustiva.
- **El siguiente ajuste:** Es igual que el "primer ajuste" con la diferencia que se deja un apuntador al lugar en donde se asignó el último hueco para realizar la siguiente búsqueda a partir de él.
- **Ajuste rápido:** Se mantienen listas ligadas separadas de acuerdo a los tamaños de los huecos, para así buscarle a los procesos un hueco más rápido en la cola correspondiente.

Otro problema que se vislumbra desde aquí es que, una vez asignado un hueco, por ejemplo, con "el peor ajuste", puede ser que el proceso requiera 12 kilobytes y que el hueco asignado fuera de 64 kilobytes, por lo cual el proceso va a desperdiciar una gran cantidad de memoria dentro de su partición, lo cual se le llama "**fragmentación interna**".

Por otro lado, conforme el sistema va avanzando en el día, finalizando procesos y comenzando otros, la memoria se va configurando como una secuencia contigua de huecos y de lugares asignados, provocando que existan huecos, por ejemplo, de 12 k, 28k y 30 k, que sumados dan 70k, pero que si en ese momento llega un

proceso pidiéndolos, no se le pueden asignar ya que no son localidades contiguas de memoria (a menos que se realice la compactación).

Al hecho de que aparezcan huecos no contiguos de memoria se le llama “**fragmentación externa**”.

De cualquier manera, la multiprogramación fue un avance significativo para el mejor aprovechamiento de la unidad central de procesamiento y de la memoria misma, así como dio pie para que surgieran los problemas de asignación de memoria, protección y relocalización, entre otros.

2.3.4 Los overlays

Una vez que surgió la multiprogramación, los usuarios comenzaron a explorar la forma de ejecutar grandes cantidades de código en áreas de memoria muy pequeñas, auxiliados por algunas llamadas al sistema operativo. Es así como nacen los “overlays”.

Esta técnica consiste en que el programador divide lógicamente un programa muy grande en secciones que puedan almacenarse en las particiones de RAM. Al final de cada sección del programa (o en otros lugares necesarios) el programador insertaba una o varias llamadas al sistema con el fin de descargar la sección presente de RAM y cargar otra, que en ese momento residía en disco duro u otro medio de almacenamiento secundario.

Aunque esta técnica era eficaz (porque resolvía el problema) no era eficiente (ya que no lo resolvía de la mejor manera).

Esta solución requería que el programador tuviera un conocimiento muy profundo del equipo de cómputo y de las llamadas al sistema operativo. Otra desventaja era la portabilidad de un sistema a otro: las llamadas cambiaban, los tamaños de particiones también.

Resumiendo, con esta técnica se podían ejecutar programas más grandes que las particiones de RAM, donde la división del código corría a cuenta del programador y el control a cuenta del sistema operativo.

2.4 Multiprogramación en memoria virtual

La necesidad cada vez más imperiosa de ejecutar programas grandes y el crecimiento en poder de las unidades centrales de procesamiento empujaron a los diseñadores de los sistemas operativos a implantar un mecanismo para ejecutar automáticamente programas más grandes que la memoria real disponible, esto es, de ofrecer “memoria virtual”.

La **memoria virtual** se llama así porque el programador ve una cantidad de memoria mucho mayor que la real, y en realidad se trata de la suma de la memoria de almacenamiento primario y una cantidad determinada de almacenamiento secundario.

El sistema operativo, en su módulo de manejo de memoria, se encarga de intercambiar programas enteros, segmentos o páginas entre la memoria real y el medio de almacenamiento secundario. Si lo que se intercambia son procesos enteros, se habla entonces de **multiprogramación en memoria real**, pero si lo que se intercambian son segmentos o páginas, se puede hablar de **multiprogramación con memoria virtual**.

La memoria virtual se apoya en varias técnicas interesantes para lograr su objetivo. Una de las teorías más fuertes es la del “conjunto de trabajo”, la cual se refiere a que un programa o proceso no está usando todo su espacio de direcciones en todo momento, sino que existen un conjunto de localidades activas que conforman el “conjunto de trabajo”.

Si se logra que las páginas o segmentos que contienen al conjunto de trabajo estén siempre en RAM, entonces el programa se desempeñará muy bien.

Otro factor importante es si los programas exhiben un fenómeno llamado “localidad”, lo cual quiere decir que algunos programas tienden a usar mucho las instrucciones que están cercanas a la localidad de la instrucción que se está ejecutando actualmente.

2.4.1 Mecanismos de asignación

Para ejecutar un proceso este debe cargarse en memoria. Generalmente el proceso reside en disco como archivo binario o ejecutable. El conjunto de procesos en disco que esperan entrar en la memoria para ejecutarse forman la cola de entrada.

El procedimiento normal consiste en seleccionar uno de los procesos de la cola de entrada y cargarlos en memoria. Esto ocasiona la relocalización de dirección o enlaces a referencias externas, según sea el caso. Mientras se ejecuta un programa, se accede a las instrucciones o datos en la memoria.

Finalmente, el programa termina de ejecutarse y su espacio en memoria se declara disponible.

Un proceso de usuario puede residir en cualquier parte de la memoria física. Por esto, aunque el espacio de direcciones de la computadora comience con 00000, no necesariamente debe de ser ocupado por el proceso del usuario.

En la mayoría de los casos, un programa de usuario pasará por varias etapas (algunas pueden ser optativas) antes de ejecutarse. En estas etapas las direcciones pueden representarse de maneras distintas.

En un programa fuente las direcciones son generalmente simbólicas. Un compilador enlazará estas direcciones simbólicas con direcciones relocalizables (como 14 bytes a partir del inicio del módulo).

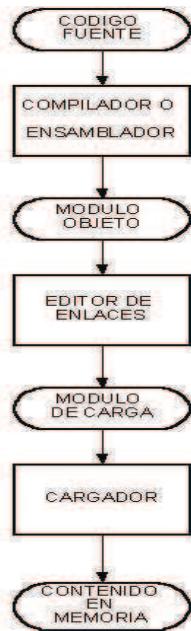
A su vez, el director de enlaces o cargador enlazará las direcciones relocalizables con direcciones absolutas (como 70414). Cada enlace es una correspondencia entre un espacio de direcciones y otro (hueco de direcciones virtuales).

El enlace de las instrucciones y datos con las instrucciones y datos con las direcciones de memoria casi siempre puede efectuarse en cualquier etapa del camino:

Compilación	Si en la compilación se sabe donde residirá el programa en memoria, se puede generar código absoluto. Es decir, si se sabe la posición donde inicia el código, entonces el compilador comenzaría en esa posición y se extendería a partir de ahí. Si se cambia la posición de inicio, será necesario volver a compilar el código. Los programas con formato .COM se enlanzan de manera absoluta durante la compilación.
Carga	Si en la compilación no se conoce dónde residirá el programa en memoria, entonces el compilador deberá cargar el código relocalizable.
Ejecución	Si durante la ejecución el proceso puede moverse de un segmento de memoria a otro, entonces el enlace final se debe posponer hasta el momento de la ejecución. Esto depende de hardware especial.

Vemos en este diagrama como se inicia a partir del código fuente de algún lenguaje de alto nivel, pasando a través de un compilador a ensamblador para crear así el código objeto, si es posible este es enlazado por el editor de enlaces.

Si las direcciones no son absolutas, es necesario que el módulo de carga realice las operaciones necesarias con el código relocalizable:

Gráfica 58. Proceso enlace de direcciones en memoria

2.4.2 Implementación de los mecanismos de asignación de memoria virtual

Existen tres estrategias de administración de la memoria virtual:

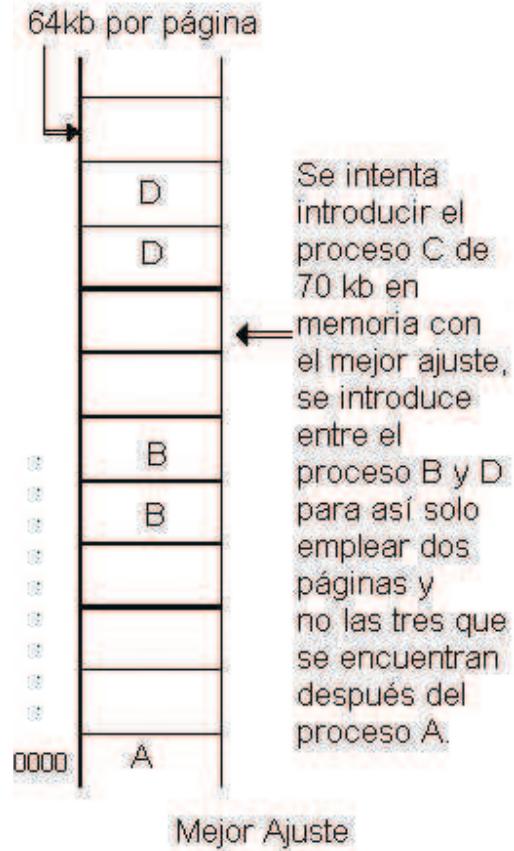
OBTENCIÓN	Determina cuando es que se debe transferir una página o segmento del almacenamiento secundario al primario. Se espera a que un proceso en ejecución haga referencia a una página o segmento antes de traer la página o segmento antes de traer la página o segmento al almacenamiento primario, y además intenta determinar por anticipado a qué lugar se hará referencia.
COLOCACIÓN	Determina en qué lugar del almacenamiento primario se debe colocar una página o segmento entrante. Esto hace que sea una decisión trivial de colocación, porque la página entrante se puede ubicar en cualquier marco de página disponible.
REEMPLAZO	Sirve para decidir cuál página o segmento se debe desplazar para dejar espacio a una página o segmento entrante cuando está ocupado el almacenamiento primario. En este caso, las rutinas de almacenamiento primario se van a desplazar para dejar espacio a una página entrante. Estrategia de colocación del almacenamiento.

Las estrategias de colocación del almacenamiento sirven para determinar en qué lugar del almacenamiento primario se deben colocar los programas y datos entrantes.

Estrategia de mejor ajuste

Un trabajo que entre en el sistema se colocará en el hueco del almacenamiento primario principal en el que quepa mejor y que deje la menor cantidad de espacios posibles sin utilizar. Para muchos, el mejor ajuste parece ser intuitivamente la estrategia más efectiva.

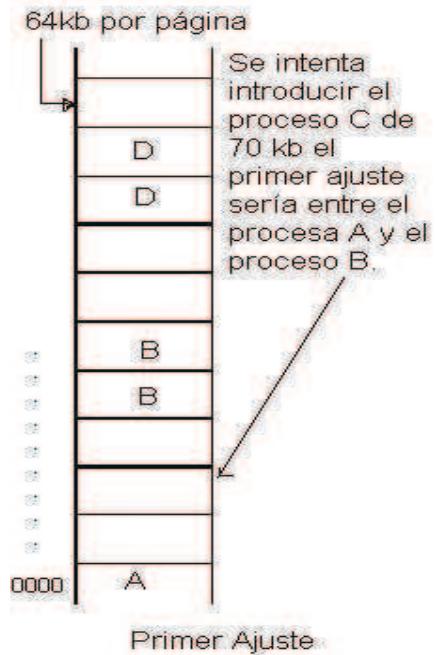
Gráfica 59. Estrategia de mejor ajuste



Estrategia de primer ajuste

Un trabajo que entre en el sistema se colocará en el almacenamiento primario principal en el primer hueco disponible, lo suficientemente grande para contenerlo.

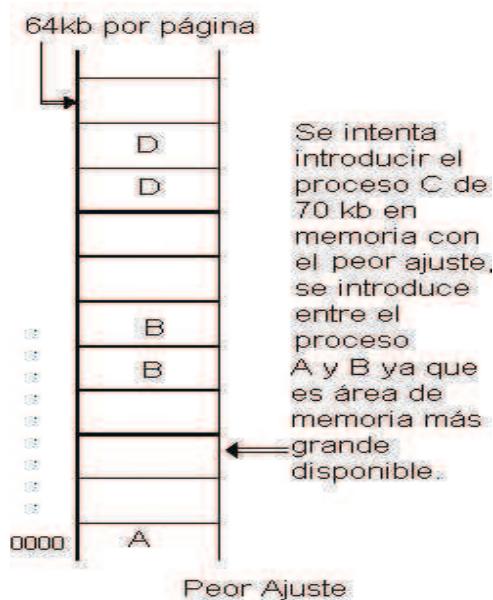
Gráfica 60. Estrategia de primer ajuste



Estrategia de peor ajuste

Consiste en colocar un programa en el almacenamiento primario en el hueco donde peor se ajusta, es decir, el hueco más grande.

Gráfica 61. Estrategia de peor ajuste



2.4.3 Paginación pura

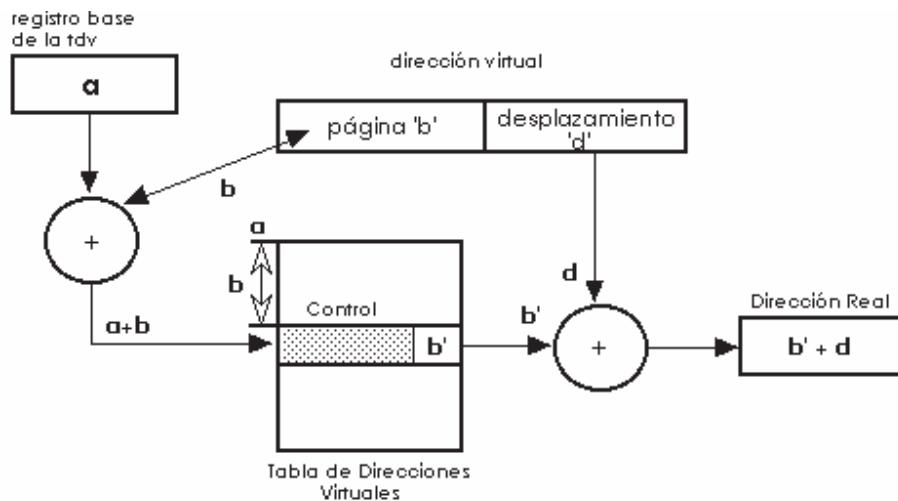
La paginación pura en el manejo de memoria consiste en que el sistema operativo divide dinámicamente los programas en unidades de tamaño fijo (generalmente múltiplos de 1 kilobyte) los cuales va a manipular de RAM a disco y viceversa.

Al proceso de intercambiar páginas, segmentos o programas completos entre RAM y disco se le conoce como “**intercambio**” o “**swapping**”.

En la paginación, se debe cuidar el tamaño de las páginas, ya que si éstas son muy pequeñas el control, por parte del sistema operativo para saber cuáles están en RAM y cuáles en disco, sus direcciones reales, etc; crece y provoca mucha “sobrecarga” (overhead). Por otro lado, si las páginas son muy grandes, el overhead disminuye pero entonces puede ocurrir que se desperdicie memoria en procesos pequeños. Debe haber un equilibrio.

Uno de los aspectos más importantes de la paginación, así como de cualquier esquema de memoria virtual, es la forma de traducir una dirección virtual a dirección real. Para explicarlo, obsérvese la siguiente gráfica:

Gráfica 62. Traducción de direcciones de memoria



Como se observa, una dirección virtual “v” = (b,d) está formada por un número de página virtual “b2 y un desplazamiento “d”.

Por ejemplo, si el sistema ofrece un espacio de direcciones virtuales de 64 kilobytes, con páginas de 4 kilobytes y la RAM sólo es de 32 kilobytes, entonces tenemos 16 páginas virtuales y 8 reales. La tabla de direcciones virtuales contiene 16 entradas, una por cada página virtual.

En cada entrada, o registro de la tabla de direcciones virtuales se almacenan varios datos: si la página está en disco o en memoria, quién es el dueño de la página, si la página ha sido modificada o es de lectura nada más, etc.

Pero el dato que nos interesa ahora es el número de página real que le corresponde a la página virtual. Obviamente, de las 16 virtuales, sólo ocho tendrán un valor de control que dice que la página está cargada en RAM, así como la dirección real de la página, denotada en la gráfica anterior como b' .

Por ejemplo, supóngase que para la página virtual número 14 la tabla dice que, efectivamente está cargada y es la página real 2 (dirección de memoria 8192).

Una vez encontrada la página real, se le suma el desplazamiento, que es la dirección que deseamos dentro de la página buscada ($b' + d$).

La tabla de direcciones virtuales a veces está ubicada en la misma memoria RAM, por lo cual se necesita saber en qué dirección comienza, en este caso, existe un registro con la dirección base denotada por la letra "a".

Cuando se está buscando una página cualquiera y ésta no está cargada, surge lo que se llama un "fallo de página" (page fault). Esto es caro para el manejador de memoria, ya que tiene que realizar una serie de pasos extra para poder resolver la dirección deseada y darle su contenido a quien lo pide. Primero, se detecta que la página no está presente y entonces se busca en la tabla la dirección de esta página en disco. Una vez localizada en disco se intenta cargar en alguna página libre de RAM. Si no hay páginas libres se tiene que escoger alguna para enviarla hacia el disco. Una vez escogida y enviada a disco, se marca su valor de control en la tabla de direcciones virtuales para indicar que ya no está en RAM, mientras que la página deseada se carga en RAM y se marca su valor para indicar que ahora ya está en RAM. Todo este procedimiento es caro, ya que se sabe que los accesos a disco duro son del orden de decenas de veces más lentos que en RAM.

En el ejemplo anterior se mencionó que cuando se necesita descargar una página de RAM hacia disco se debe de hacer una elección. Para realizar esta elección existen varios algoritmos, los cuales se describen enseguida.

La primera en entrar, primera en salir	Se escoge la página que haya entrado primero y esté cargada en RAM. Se necesita que en los valores de control se guarde un dato de tiempo. No es eficiente porque no aprovecha ninguna característica de ningún sistema. Es justa e imparcial.
La no usada recientemente	Se escoge la página que no haya sido usada (referenciada) en el ciclo anterior. Pretende aprovechar el hecho de la localidad en el conjunto de trabajo. R La usada menos recientemente: Es parecida a la anterior, pero escoge la página que se usó hace más tiempo, pretendiendo que como ya tiene mucho sin usarse es muy probable que siga sin usarse en los próximos ciclos. Necesita de una búsqueda exhaustiva.

La no usada frecuentemente	Este algoritmo toma en cuenta no tanto el tiempo, sino el número de referencias. En este caso cualquier página que se use muy poco, menos veces que alguna otra.
La menos frecuentemente usada	Es parecida a la anterior, pero aquí se busca en forma exhaustiva aquella página que se ha usado menos que todas las demás. R En forma aleatoria: Elige cualquier página sin aprovechar nada. Es justa e imparcial, pero ineficiente.

Otro dato interesante de la paginación es que ya no se requiere que los programas estén ubicados en zonas de memoria adyacente, ya que las páginas pueden estar ubicadas en cualquier lugar de la memoria RAM.

Tomando en consideración que la paginación trabaja con memoria virtual, entonces recordaremos que en la memoria virtual se encuentran las direcciones virtuales, las cuales se generan por índices, registros, palabras o bytes. Estas direcciones virtuales no pasan de forma directa al bus de memoria, sino que van a la unidad de administración de memoria (MMU), y por medio de un chip se asocian las virtuales con las reales.

La paginación surge de la necesidad de crear espacios de memoria contiguos, pues debido a que la fragmentación genera espacios dispersos de almacenamiento, no se pueden ejecutar los procesos.

Claro está que esto se soluciona mediante la paginación y la compactación. La paginación permite que la memoria de un proceso no sea contigua (almacenamiento secundario), pero al ser asignado al almacenamiento primario necesariamente sería contigua.

El usuario ya no se preocupa al tener espacio disponible en el almacenamiento secundario, pues así se lograrán ejecutar los procesos. La paginación es un método común en los sistemas operativos.

La memoria física se divide en bloques de tamaño fijo llamados **Marcos**.

La memoria lógica se divide en bloques del mismo tamaño **páginas**.

Cuando un proceso se va a ejecutar, sus páginas se cargan desde el almacenamiento auxiliar(secundario) en cualquiera de los marcos disponibles. Por lo tanto al crear la paginación (páginas) se crean también los marcos de las páginas resultantes del número de páginas que ocurran. Viendo entonces que la paginación se da en el almacenamiento secundario (disco, dispositivos), estamos haciendo uso del hardware por lo que el hardware representa el apoyo para la paginación.

2.4.4 Segmentación pura

La segmentación se aprovecha del hecho de que los programas se dividen en partes lógicas, como son las partes de datos, de código y de pila (stack). La segmentación asigna particiones de memoria a cada segmento de un programa y busca como objetivos el hacer fácil el compartir segmentos (por ejemplo librerías compartidas) y el intercambio entre memoria y los medios de almacenamiento secundario.

En la segmentación pura las particiones de memoria son de tamaño variable, en contraste con páginas de tamaño fijo en la paginación pura.

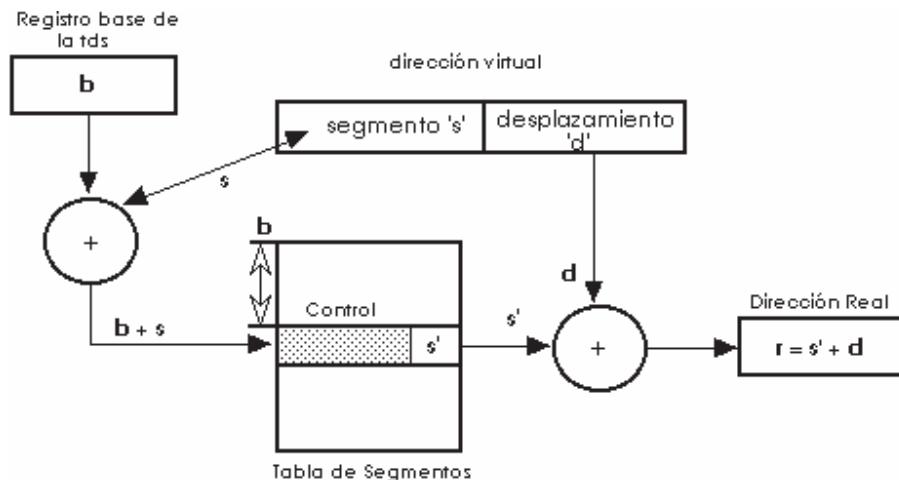
También se puede decir que la segmentación pura tiene una granularidad menor que la paginación por el tamaño de segmentos versus tamaño de páginas.

Nuevamente, para comprender mejor la segmentación, se debe dar un repaso a la forma en que las direcciones virtuales son traducidas a direcciones reales (Gráfica 63).

Prácticamente la traducción es igual que la llevada a cabo en la paginación pura, tomando en consideración que el tamaño de los bloques a controlar por la tabla de traducción son variables, por lo cual, cada entrada en dicha tabla debe contener la longitud de cada segmento a controlar. Otra vez se cuenta con un registro base que contiene la dirección del comienzo de la tabla de segmentos. La dirección virtual se compone de un número de segmento (s) y un desplazamiento (d) para ubicar un byte (o palabra) dentro de dicho segmento. Es importante que el desplazamiento no sea mayor que el tamaño del segmento, lo cual se controla simplemente chequeando que ese valor sea mayor que la dirección del inicio del segmento y menor que el inicio sumado al tamaño.

Una vez dada una dirección virtual $v=(s,d)$, se realiza la operación $b + s$ para hallar el registro (o entrada de la tabla de segmentos) que contiene la dirección de inicio del segmento en la memoria real, denotado por s' .

Ya conociendo la dirección de inicio en memoria real s' sólo resta encontrar el byte o palabra deseada, lo cual se hace sumándole a s' el valor del desplazamiento, de modo que la dirección real $r = s' + d$.

Gráfica 63. Traducción de direcciones de memoria en segmentación

La finalidad de la tabla de páginas es asociar páginas virtuales con marcos de página.

La entrada típica en una tabla de páginas es la siguiente:



- **La dirección del marco de página**, guarda la dirección en memoria real donde se encuentra la página.
- **El bit presente/ausente**, indica si la página reside en la memoria real o no. Si la página no está presente (0) y se referencia, se producirá un defecto de página.
- **Los bits de protección** indican el tipo de acceso permitido. Estos generalmente se conocen como ***rwx*** y dependiendo de si están encendidos, la página se podrá leer (***r***), escribir (***w***) o ejecutar (***x***).
- **El bit de modificación** se enciende cada que la página sufre algún cambio en memoria real. Se utiliza cuando la página es seleccionada para salir de la memoria real, cuando se presenta un defecto de página. Si el bit está

encendido, entonces la página se escribirá sobre el disco. En caso contrario sólo se desecha pues la imagen en disco es igual a la existente en la memoria real.

- **El bit de referencia** como su nombre lo indica se enciende cada vez que la página es referenciada, bien sea para lectura o escritura. Sirve para apoyar al sistema operativo, para seleccionar la página a salir cuando se presenta un defecto de página. Las páginas no referenciadas son unas muy buenas candidatas a salir.
- **El bit de caching.** Si esté esta desactivado, cuando se produzca una operación de E/S, el sistema operativo busca la información en el hardware y no una copia antigua en el caché.

Cada entrada en la tabla de segmentos tiene un formato similar al mostrado en la siguiente tabla. Se tienen campos que indican la longitud, los permisos, la presencia o ausencia y dirección de inicio en memoria real del segmento.

0 está 1 no está			lectura escritura ejecución adición permisos	
bit de residencia	dirección en disco	longitud		dirección real

Un hecho notable en los sistemas que manejan paginación es que cuando el proceso comienza a ejecutarse ocurren un gran número de fallos de página, porque es cuando está referenciando muchas direcciones nuevas por primera vez, después el sistema se estabiliza, conforme el número de marcos asignados se acerca al tamaño del conjunto de trabajo.

2.4.5 Sistemas combinados

La paginación y la segmentación puras son métodos de manejo de memoria bastante efectivos, la mayoría de los sistemas operativos modernos implantan esquemas combinados, es decir, combinan la paginación y la segmentación.

La idea de combinar estos esquemas se debe a que de esta forma se aprovechan los conceptos de la división lógica de los programas (segmentos) con la granularidad de las páginas. De esta forma, un proceso estará repartido en la memoria real en pequeñas unidades (páginas) cuya liga son los segmentos.

También es factible así el compartir segmentos a medida que las partes necesitadas de los mismos se van referenciando (páginas). Para comprender este esquema, nuevamente se verá cómo se traduce una dirección virtual en una localidad de memoria real. Para la paginación y segmentación puras se puede decir que el direccionamiento es “bidimensional” porque se necesitan dos valores

para hallar la dirección real. Para el caso combinado, se puede decir que se tiene un direccionamiento “tridimensional”.

En la gráfica 64 se muestran las partes relevantes para lograr la traducción de direcciones. El sistema debe contar con una tabla de procesos (TP).

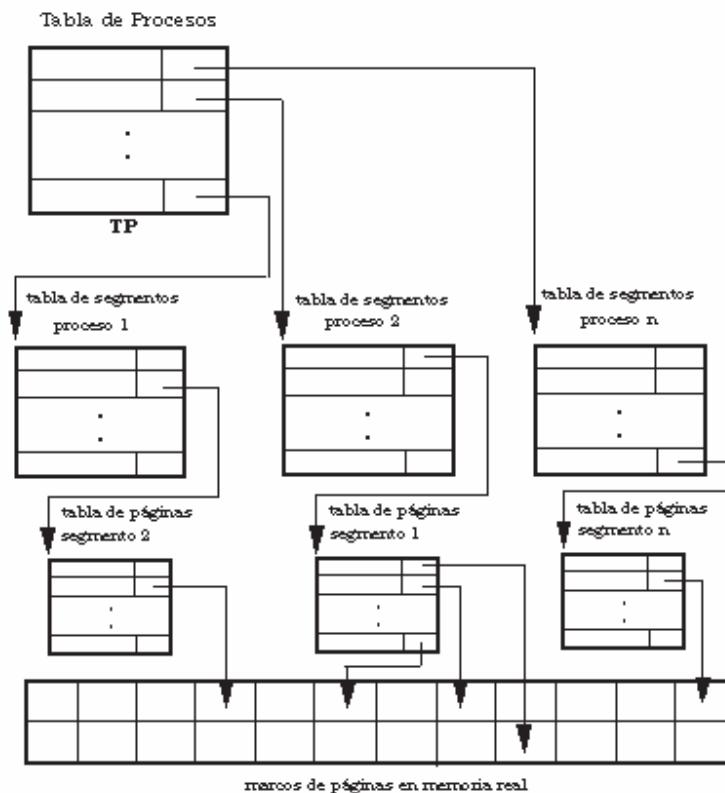
Por cada renglón de esa tabla se tiene un número de proceso y una dirección a una tabla de segmentos. Es decir, cada proceso tiene una tabla de segmentos.

Cuando un proceso hace alguna referencia a memoria, se consulta TP para encontrar la tabla de segmentos de ese proceso. En cada tabla de segmentos de proceso (TSP) se tienen los números de los segmentos que componen a ese proceso. Por cada segmento se tiene una dirección a una tabla de páginas. Cada tabla de páginas tiene las direcciones de las páginas que componen a un solo segmento.

Por ejemplo, el segmento “A” puede estar formado por las páginas reales “a”, “b”, “c”, “p” y “x”. El segmento “B” puede estar compuesto de las páginas “f”, “g”, “j”, “w” y “z”.

Para traducir una dirección virtual $v=(s,p,d)$ donde “s” es el segmento, “p” es la página y “d” el desplazamiento, en la página se hace lo siguiente:

Primero se ubica de qué proceso es el segmento y se localiza la tabla de segmentos de ese proceso en la TP. Con s' como índice se encuentra un renglón (registro) en la tabla de segmentos de ese proceso y en ese renglón está la dirección de la tabla de páginas que componen al segmento. Una vez en la tabla de páginas, se usa el valor p' como índice para encontrar la dirección de la página en memoria real. Una vez en esa dirección de memoria real, se encuentra el byte (o palabra) requerido por medio del valor de d' .

Gráfica 64. Traducción en la segmentación – paginación

Ahora, en este esquema pueden haber dos tipos de fallos: por fallo de página y por fallo de segmento.

Cuando se hace referencia a una dirección y el segmento que la contiene no está en RAM (aunque sea parcialmente), se provoca un fallo por falta de segmento y lo que se hace es traerlo del medio de almacenamiento secundario y crearle una tabla de páginas. Una vez cargado el segmento se necesita localizar la página correspondiente, pero ésta no existe en RAM, por lo cual se provoca un fallo de página y se carga de disco y finalmente se puede ya traer la dirección deseada por medio del desplazamiento de la dirección virtual.

La eficiencia de la traducción de direcciones tanto en paginación pura, segmentación pura y esquemas combinados se mejora usando memorias asociativas para las tablas de páginas y segmentos, así como memorias caché para guardar los mapeos más solicitados.

Otro aspecto importante es la estrategia para cargar páginas (o segmentos) a la memoria RAM. Se usan más comúnmente dos estrategias: **cargado de páginas por demanda** y **cargado de páginas anticipada**.

La estrategia de **cargado por demanda** consiste en que las páginas solamente son llevadas a RAM si fueron solicitadas, es decir, si se hizo referencia a una dirección que cae dentro de ellas.

La **carga anticipada** consiste en tratar de adivinar qué páginas serán solicitadas en el futuro inmediato y cargarlas de antemano, para que cuando se pidan ya no ocurran fallos de página.

2.5 Algoritmos de reemplazo de páginas

Existen muchos y diferentes algoritmos para el reemplazo de página, pero en general se prefieren los que presenten el menor número de fallos de página,

Los fallos de página, sólo ocurren en dos ocasiones:

- Cuándo la memoria está vacía y entran solo las páginas disponibles de acuerdo al tamaño del marco de páginas.
- Cuando entran las demás páginas en localidades diferentes a la página que entra en ese momento.

Para realizar esta elección existen varios algoritmos, los cuales se describen enseguida.

La primera en entrar, primera en salir: Se escoge la página que haya entrado primero y esté cargada en RAM. Se necesita que en los valores de control segura de un dato de tiempo. No es eficiente porque no aprovecha ninguna característica de ningún sistema. Es justa e imparcial.

La no usada recientemente. Se escoge la página que no haya sido usada (referenciada) en el ciclo anterior. Pretende aprovechar el hecho de la localidad en el conjunto de trabajo.

La usada menos recientemente. Es parecida a la anterior, pero escoge la página que se usó hace más tiempo, pretendiendo que como ya tiene mucho sin usarse es muy probable que siga sin usarse en los próximos ciclos. Necesita de una búsqueda exhaustiva.

La no usada frecuentemente. Este algoritmo toma en cuenta no tanto el tiempo, sino el número de referencias. En este caso cualquier página que se use muy poco, menos veces que alguna otra.

La menos frecuentemente usada. Es parecida a la anterior, pero aquí se busca en forma exhaustiva aquella página que se ha usado menos que todas las demás.

En forma aleatoria. Elige cualquier página sin aprovechar nada. Es justa e imparcial, pero ineficiente. Otro dato interesante de la paginación es que ya no se requiere que los programas estén ubicados en zonas de memoria adyacente, ya que las páginas pueden estar ubicadas en cualquier lugar de la memoria RAM.

2.5.1 Algoritmo FIFO

El algoritmo de remplazo de páginas más sencillo es el primero en entrar, primero en salir (First Input, first output). Un algoritmo FIFO asocia a cada página el instante en el que se trajo a memoria. Cuando hay que reemplazar una página se elige la más antigua.

Si construimos una cola FIFO que contenga las en memoria reemplazamos la página de inicio de la cola y cuando se introduce en memoria una página, la insertamos al final de la cola.

Por ejemplo: Teniendo la serie de referencias con un marco de página de tamaño 3, observamos: 7 0 1 2 3 0 4 2 3 0 1 2 7 0 1

7	7	7	2	2	2	4	4	4	0	0	0	7	7	7
0	0	0	3	3	3	2	2	2	1	1	1	0	0	0
1	1	1	0	0	0	3	3	3	2	2	2	1	1	1

Nota: Los cuadros en color gris indican las páginas que entran.

Después de descargar una página activa para incorporar una página nueva, se provocará casi de inmediato una falla por la página activa. Tendremos que reemplazar otra página para devolver memoria la página activa. De esta manera, una mala elección en el reemplazo aumenta la cantidad de fallos de páginas y frena la ejecución de procesos, pero no provoca ejecución incorrecta.

Al elegir una página equivocada el sistema operativo tiene varias opciones conocidas como **ANOMALIA DE BELADY**.

1. Abortar el proceso del usuario lo cual no es la mejor opción ya que el sistema operativo realiza la paginación para mejorar la utilización y productividad el sistema de computadora.
2. Descargar un proceso de páginas por cualquier algoritmo de reemplazo.
3. Reemplazo de páginas por cualquier algoritmo de reemplazo.

En este ejemplo se muestra como se implementan los algoritmos de reemplazo de páginas. Tomamos como referencia la siguiente serie:

9 0 1 8 3 1 0 4 6 3 8 7 0 8 2 5 1 3 4 5 0 7 1 2 8 4 6 7 1 2 0 9.

1	1	1	1	0	0	0	3	3	3	0	0	0	1	1	1	5	5	5	1	1	1	4	4	4	1	1	1	9
0	0	3	3	3	3	6	6	6	7	7	7	7	5	5	5	4	4	4	7	7	7	8	8	8	7	7	7	0
9	8	8	8	8	4	4	4	8	8	8	8	2	2	2	3	3	3	3	0	0	0	2	2	2	6	6	6	2

Total de fallos = 30

Nota: Los cuadros en color gris indican las páginas que entran.

2.5.2 Algoritmo óptimo

Una consecuencia del estudio de la anomalía de Belady, fue la búsqueda de un algoritmo de reemplazo de páginas óptimo. Este algoritmo tiene la menor tasa de fallos de página de todos los algoritmos: un algoritmo óptimo nunca presentará la anomalía de Belady.

Consiste en reemplazar la página que no se usará durante el mayor periodo de tiempo. Por ejemplo considerando la siguiente serie de referencias, tendremos que: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

7	7	7	2	2	2	2	2	7
0	0	0	0	4	0	0	0	
	1	1	3	3	3	1	1	

Nota: Los cuadros en negrillas indican las páginas que entran.

Las primeras tres referencias causan fallas que llenan los tres marcos vacíos. La referencia a la página 2 reemplaza a la página 7 hasta la referencia número 1, la página 0 se usará en la página 5 y la página 1 en la referencia No. 14. La referencia a la página 3 reemplaza a la página 1, ya que esta será la última de las 3 páginas en memoria a la que se volverá a hacer referencia. Con sólo 9 fallos de página el reemplazo óptimo es mucho mejor que el FIFO que con la misma serie presenta 15 fallos de página. De hecho, ningún algoritmo de reemplazo puede procesar esta serie de referencias con menos de 9 fallos usando un marco de tamaño 3. Por desgracia, es difícil de implantar el algoritmo de reemplazo de páginas óptimo, ya que se requiere el conocimiento a futuro de la serie de referencias.

2.5.3 Algoritmo LRU

Si el algoritmo no es tan factible quizá sea posible una aproximación. La principal diferencia entre el FIFO y el OPTIMO, es que el FIFO utiliza el instante en que entró la página a memoria; el OPTIMO utiliza el tiempo en que usará la página.

Si utilizamos el pasado reciente como aproximación de que sucederá en un futuro cercano, reemplazaremos la página que no se ha utilizado durante el mayor periodo de tiempo posible.

Este es el algoritmo MENOS RECIENTE USADO (LRU, Least Recently Used). El reemplazo LRU asocia a cada página al instante en que se usó por última vez.

Cuando hay que reemplazar una página, LRU asocia la página que no ha sido utilizada durante el mayor período de tiempo posible, es decir, ve hacia atrás en el tiempo (su utilización en el pasado). Por ejemplo, si tenemos la misma serie de referencias que en el óptimo 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1.

7	7	7	2	2	4	4	4	0	1	1	1
0	0	0	0	0	0	3	3	3	0	0	
	1	1	3	3	2	2	2	2	2	7	

Nota: Los cuadros en negrillas indican las páginas que entran.

El algoritmo LRU produce 12 fallos. Las cinco primeras son las mismas que en el óptimo, sin embargo, cuando se presenta la referencia a la página 4, el reemplazo observa que de las 3 páginas en memoria, la 2 fue la menos reciente usada.

La página que se usó más recientemente es la página 0 y justo antes se usó la página 3. Por lo tanto, el LRU reemplaza a la página 2, sin saber que está a punto de usarse. Luego, cuando se dé la falla por la página 2, el algoritmo LRU puede reemplazar la página 3.

A pesar de estos problemas, el reemplazo por LRU con 12 fallos sigue siendo mejor que el FIFO con 15 fallos.

La política de implantar el algoritmo LRU es generalmente buena, sin embargo existe el problema de cómo implantar el reemplazo de páginas. El problema consiste en determinar un orden para los marcos definidos por el instante del último uso. El algoritmo LRU se puede implementar de dos maneras:

Contadores. A cada entrada de la tabla de páginas asociamos un registro de instante de uso y añadimos al CPU un reloj lógico o contador. El reloj se incrementa con cada referencia a memoria. De esta manera siempre tendremos el tiempo de la última referencia a la página y reemplazaremos la página con menor valor.

Pilas. Cuando se hace referencia a una página, se saca la pila y se coloca en la parte superior. De esta manera, en la parte superior de la pila siempre se encuentra la pila más recientemente usada, y en la parte inferior de la pila la

página menos reciente usada. Puesto que las entradas deben sacarse de la mitad de la pila, puede implantarse mejor con una lista doblemente ligada.

La siguiente serie muestra el algoritmo : 9 0 1 8 3 1 0 4 6 3 8 7 0 8 2 5 1 3 4 5 0 7 1 2 8 4 6 7 1 2 0 9.

1	1	1	1	1	1	6	6	6	7	7	7	2	2	2	3	3	3	0	0	0	2	2	2	6	6	6	2	2	2
0	0	3	3	3	4	4	4	8	8	8	8	8	8	1	1	1	5	5	5	1	1	1	4	4	4	1	1	1	9
9	8	8	8	0	0	0	3	3	3	0	0	0	5	5	5	4	4	4	7	7	7	8	8	8	8	8	8	0	0

Total de fallos = 29

Nota: Los cuadros en color gris indican las páginas que entran.

2.5.4 Algoritmos aproximados (bit de referencia)

Pocos sistemas proporcionan suficiente ayuda del hardware para un verdadero reemplazo de páginas LRU. Algunos sistemas no ofrecen ningún apoyo del hardware, por lo que hay que emplear otros algoritmos de reemplazo de páginas, sin embargo, muchos sistemas ofrecen cierta ayuda en la forma de un bit de referencia.

El hardware coloca un 1 en el bit de referencia para una página cada vez que se hace referencia a ella. Los bits de referencia están asociados a cada entrada de la tabla de páginas.

2.5.5 Algoritmo de bits adicionales de referencia

Podemos obtener información del uso de las páginas en un algoritmo de reemplazo. Esta información se puede implementar utilizando una tabla adicional de 8 bits para cada página. Estos registros contienen la historia de utilización de la página durante los 8 períodos. Si el registro del desplazamiento contiene 00000000, entonces la página no se ha utilizado por lo menos en 8 períodos. Una página que se utiliza cuando menos una vez cada periodo tendría un valor de 1 1 1 1 1 1 1 1 en su registro adicional de referencia al desplazamiento de la página. Una página con valor de 1 1 0 0 0 1 0 0, si interpretamos esto en bytes de 8 bits como enteros sin signo, la página con el menor número de 1's a la izquierda es la página menos reciente usada y puede entonces reemplazarse. Por lo mismo, puede haber más de una página que, casi no se esté usando, entonces habría que aplicar una selección FIFO entre ellas.

Por ejemplo:

PROCESO	
El proceso más usado = 2	0
El proceso menos usado = 9	1
El proceso con mayor demanda = 2	1
El proceso con menor demanda = 8	0
El proceso nunca usado = 3,	0
5	0
	0
	0
	0
	0
	0
	0
	0
	0
	0
	1
	1
	0
	1
	1
	1
	1
	0
	0
	0
	0
	1
	1
	2
	1
	1
	1
	0
	0
	0
	0
	1
	1
	1
	1
	1
	3
	0
	0
	0
	0

	0
	0
	0
	0
	4
	0
	0
	0
	0
	1
	1
	1
	1
	0
	5
	0
	0
	0
	0
	0
	0
	0
	6
	0
	0
	0
	0
	1
	1
	1
	1
	0
	0
	7
	0
	0
	1
	1
	1
	0
	0
	0

	8
	0
	0
	0
	0
	0
	1
	1
	1
	9
	1
	0
	0
	0
	0
	0
	0
	0

2.5.6 Algoritmo de segunda oportunidad

El algoritmo básico para el reemplazo de segunda oportunidad es el FIFO, sin embargo, cuando se selecciona una página examinamos su bit de referencia; si el bit dereferencia es igual a 0, reemplazamos.

Sin embargo, si el bit de referencia es 1, damos a la página la segunda oportunidad y pasa a seleccionar la siguiente página en el orden FIFO.

Cuando a una página se le da segunda oportunidad, se borra su bit de referencia y se establece como llegada actual, de esa página. De esta manera, una de las páginas a las que se les brinde una segunda oportunidad no serán reemplazadas hasta que todas las demás páginas se reemplacen (o que se les otorgue una segunda oportunidad). Además si una página se usa con frecuencia suficiente para mantener su bit de referencia en 1, nunca será reemplazada.

Actividad final:

Realice un mapa conceptual o cuadro sinóptico en donde se evidencien los diferentes algoritmos de reemplazo de páginas y los algoritmos para el manejo de memoria virtual, también paginación, segmentación y el sistema combinado.

Revise los conceptos que relacionó en la actividad inicial, compárelos con los vistos en el capítulo y concluya.

CAPÍTULO 3. ADMINISTRACIÓN DE DISPOSITIVOS²⁶

Actividad inicial:

Realice una lista de todos los dispositivos que considere hacen parte de un sistema computacional.

Y para cada uno de ellos diga si es dispositivo de entrada o de salida al sistema.

Además imagine qué tipo de controlador de proceso tiene cada dispositivo.

El código destinado a manejar la entrada y salida de los diferentes periféricos en un sistema operativo es de una extensión considerable y sumamente complejo.

Resuelve necesidades de sincronizar, atrapar interrupciones y ofrecer llamadas al sistema para los programadores. En este capítulo se repasarán los principios más importantes a tener en cuenta en el manejo de E/S.

3.1 Entrada/salida

Las funciones más importantes de un sistema computacional son procesamiento y E/S. En la mayoría de los casos, lo más importante es la E/S y el cálculo es incidental. Ejemplo:

- Navegación por Internet
- Multimedia
- Edición, etc.

En este capítulo discutiremos:

- Fundamentos del hardware de E/S
- Servicios de E/S que proporciona el S.O.
- Interfaz de software

3.1.1 Características de la E/S

La tecnología de E/S está sometida a dos fuerzas:

- Estandarización del software (SW) e interfaces de hardware (HW).
- Gran variedad de dispositivos distintos con fuerte penetración en el mercado.

²⁶Las gráficas de este capítulo fueron copiadas del documento pdf: Sistemas Operativos. Profesor Javier Cañas. Capítulo 11.

Para encapsular los detalles de los dispositivos, el S.O. utiliza módulos llamados *device drivers*.

3.2 Dispositivos de Entrada/Salida

Los dispositivos de entrada salida se dividen, en general, en dos tipos:

- Dispositivos orientados a bloques y
- Dispositivos orientados a caracteres.

Los dispositivos orientados a bloques tienen la propiedad de que se pueden direccionar, esto es, el programador puede escribir o leer cualquier bloque del dispositivo realizando primero una operación de posicionamiento sobre el dispositivo.

Los dispositivos más comunes orientados a bloques son los discos duros, la memoria, discos compactos y, posiblemente, unidades de cinta.

Por otro lado, los dispositivos orientados a caracteres son aquellos que trabajan con secuencias de bytes sin importar su longitud ni ninguna agrupación en especial. No son dispositivos direccionables.

Ejemplos de estos dispositivos son el teclado, la pantalla o display y las impresoras.

La clasificación anterior no es perfecta, porque existen varios dispositivos que generan entrada o salida que no pueden englobarse en esas categorías. Por ejemplo, un reloj que genera pulsos. Sin embargo, aunque existan algunos periféricos que no se puedan categorizar, todos están administrados por el sistema operativo por medio de una parte electrónica - mecánica y una parte de software.

Los dispositivos de E/S son muy diversos. Para poder clasificarlos se utilizan 3 características:

- **Comportamiento.** Entrada, salida o almacenamiento.
- **Contraparte.** Máquina o ser humano.
- **Tasa de transferencia.** Tasa peak de transferencia entre dispositivo y memoria.

Gráfica 65. Ejemplos de dispositivos

Dispositivo	Comportamiento	Contraparte	Tasa [KB/seg]
Teclado	Entrada	Humano	0.01
Mouse	Entrada	Humano	0.02
Voz (entrada)	Entrada	Humano	0.02
Scanner	Entrada	Humano	400
Voz (salida)	Salida	Humano	0.6
Impresora linea	Salida	Humano	1.0
Impresora laser	Salida	Humano	200
Display Gráfico	Salida	Humano	60000
Modem Red LAN	Entrada/salida Entrada/salida	Máquina Máquina	2.0 - 8.0 500 - 6000
Floppy	Almacenamiento	Máquina	100
CD	Almacenamiento	Máquina	1000
Cinta magnética	Almacenamiento	Máquina	2000
Disco magnético	Almacenamiento	Máquina	2000-10000

3.3 Controladores de Dispositivos (Terminales y Discos Duros)

Los controladores de dispositivos (también llamados adaptadores de dispositivos) son la parte electrónica de los periféricos, pueden tener la forma de una tarjeta o un circuito impreso integrado a la tarjeta maestra de la computadora.

Por ejemplo, existen controladores de discos que se venden por separado y que se insertan en una ranura de la computadora, o existen fabricantes de computadoras que integran esa funcionalidad en la misma tarjeta en que viene la unidad central de procesamiento (tarjeta maestra).

Los controladores de dispositivos generalmente trabajan con voltajes de 5 y 12 volts con el dispositivo propiamente, y con la computadora a través de interrupciones. Estas interrupciones viajan por el “bus” de la computadora y son recibidos por la CPU la cual a su vez pondrá en ejecución algún programa que sabrá qué hacer con esa señal. A ese programa se le llama “manejador de dispositivo” (device driver).

Algunas veces el mismo controlador contiene un pequeño programa en una memoria de solo lectura o en memoria de acceso aleatorio no volátil y re-escribible que interactúa con el correspondiente manejador en la computadora.

Para intercambiar datos o señales entre la computadora y los controladores, muchas veces se usan registros o secciones predefinidas de la memoria de la computadora. A este esquema se le llama “manejo de entrada - salida mapeado por memoria” (memory mapped I/O).

3.4 Principios en el Software de Entrada - Salida

Los principios de software en la entrada - salida se resumen en cuatro puntos: el software debe ofrecer manejadores de interrupciones, manejadores de dispositivos, software que sea independiente de los dispositivos y software para usuarios.

3.4.1 Manejadores de interrupciones

El primer objetivo referente a los manejadores de interrupciones consiste en que el programador o el usuario no debe darse cuenta de los manejos de bajo nivel para los casos en que el dispositivo está ocupado y se debe suspender el proceso o sincronizar algunas tareas. Desde el punto de vista del proceso o usuario, el sistema simplemente se tardó más o menos en responder a su petición.

3.4.2 Manejadores de dispositivos

El sistema debe proveer los manejadores de dispositivos necesarios para los periféricos, así como ocultar las peculiaridades del manejo interno de cada uno de ellos, tales como el formato de la información, los medios mecánicos, los niveles de voltaje y otros. Por ejemplo, si el sistema tiene varios tipos diferentes de discos duros, para el usuario o programador las diferencias técnicas entre ellos no le deben importar, y los manejadores le deben ofrecer el mismo conjunto de rutinas para leer y escribir datos.

3.4.3 Software independiente del dispositivo

Este es un nivel superior de independencia que el ofrecido por los manejadores de dispositivos. Aquí el sistema operativo debe ser capaz, en lo más posible, de ofrecer un conjunto de utilerías para accesar periféricos o programarlos de una manera consistente. Por ejemplo, que para todos los dispositivos orientados a bloques se tenga una llamada para decidir si se desea usar "buffers" o no, o para posicionarse en ellos.

3.4.4 Software para usuarios

La mayoría de las rutinas de entrada - salida trabajan en modo privilegiado, o son llamadas al sistema que se ligan a los programas del usuario formando parte de sus aplicaciones y que no le dejan ninguna flexibilidad al usuario en cuanto a la apariencia de los datos.

Existen otras librerías en donde el usuario si tiene poder de decisión (por ejemplo la llamada a "printf" en el lenguaje "C").

Otra facilidad ofrecida son las áreas de trabajos encolados (spooling areas), tales como las de impresión y correo electrónico.

3.5 Reloj

Los relojes son esenciales para el buen funcionamiento de cualquier sistema porque juegan un papel decisivo en la sincronización de procesos, en la calendarización de trabajos por lote y para la asignación de turnos de ejecución entre otras tareas relevantes.

Generalmente se cuenta con dos relojes en el sistema: uno que lleva la hora y fecha del sistema y que oscila entre 50 y 60 veces por segundo y el reloj que oscila entre 5 y 100 millones de veces por segundo y que se encarga de enviar interrupciones a la CPU de manera periódica.

El reloj de mayor frecuencia sirve para controlar el tiempo de ejecución de los procesos, para despertar los procesos que están “durmiendo” y para lanzar o iniciar procesos que fueron calendarizados.

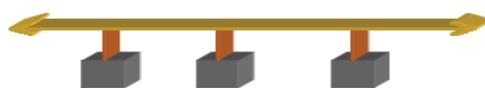
Para mantener la hora y fecha del sistema generalmente se usa un registro alimentado por una pila de alta duración que almacena estos datos y que se programan de fábrica por primera vez. Así, aunque se suspenda la energía la fecha permanece.

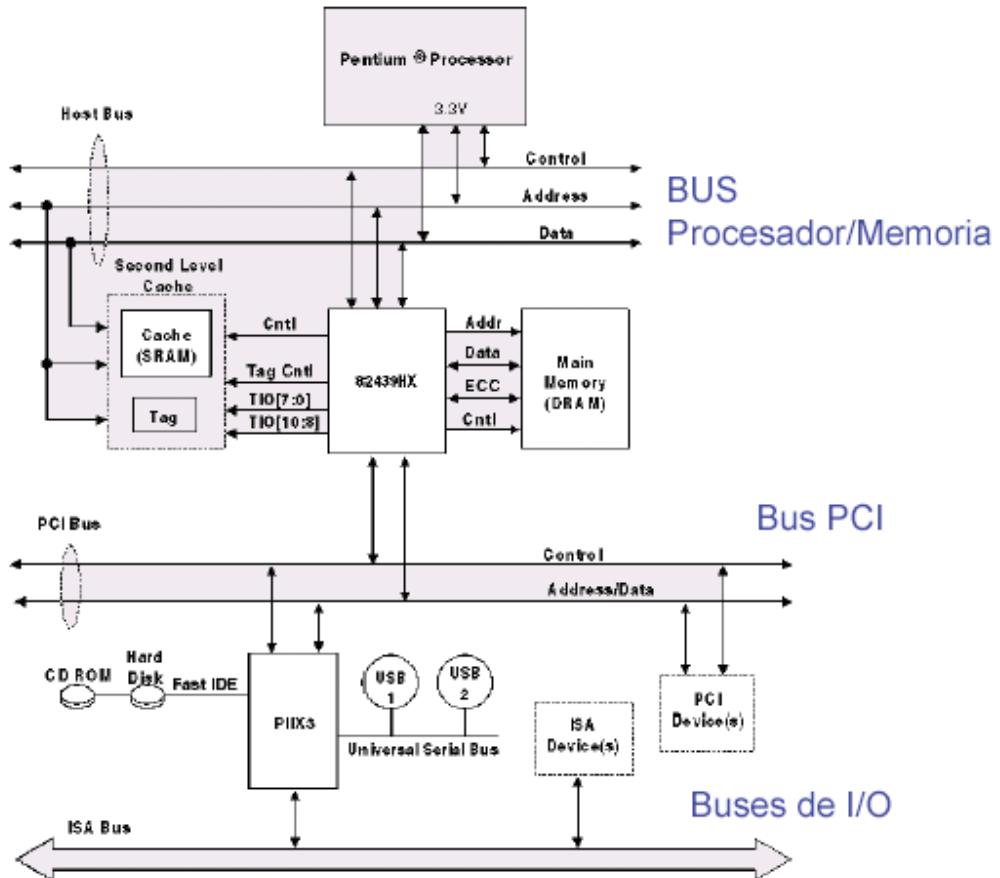
Para lanzar procesos (chequeo de tiempo ocioso de un dispositivo, terminación del time slice de un proceso, etc), se almacena un valor en un registro (valor **QUANTUM**) el cual se decrementa con cada ciclo del reloj, y cuando llega a cero se dispara un proceso que ejecutará las operaciones necesarias (escoger un nuevo proceso en ejecución, verificar el funcionamiento del motor del disco flexible, hacer eco de un carácter del teclado, etc).

3.6 Puertas y buses

Un dispositivo se conecta con un sistema computacional a través de puertas y buses.

Un Bus se utiliza para conectar diversos subsistemas: CPU - Memoria - E/S. Un Bus es un **medio compartido**.



Gráfica 66. Organización sistema pentium. Buses

3.6.1 Manejo de Controladores

¿Cómo el procesador envía comandos y datos a los controladores de dispositivos?

Los controladores tienen registros para datos y control. El procesador envía bits a estos registros.

Para esto se usan dos estrategias:

- Instrucciones especiales de E/S
- Mapa de memoria: los registros del controlador están en el espacio de direcciones del procesador.

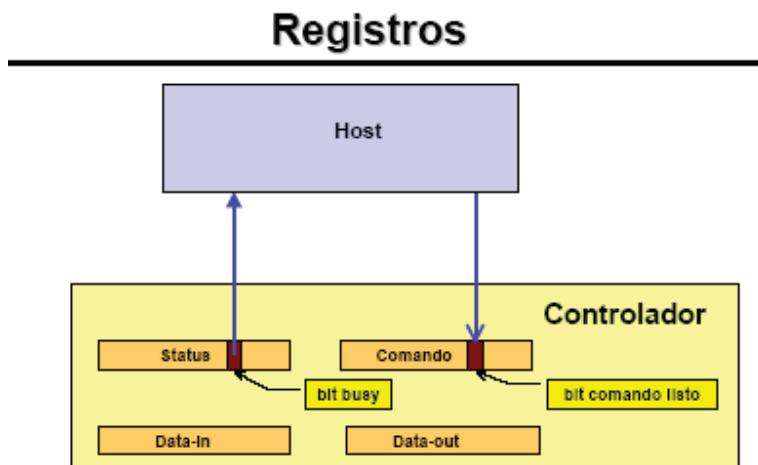
Una puerta de E/S tiene 4 registros: ***status*, *control*, *data-in* y *data-out***.

3.7 Protocolos de interacción

Los protocolos de interacción en su detalle son bastante complicados, pero en lo esencial son simples:

- El controlador y el host establecen una relación productor-consumidor.
- Se usan dos bits en el controlador para coordinar la interacción:
 - **busy**: bit del registro de status que indica que el controlador está ocupado (con un 1)
 - **command-ready**: bit del registro de comando, activado por el host

Gráfica 67. Coordinación de la interacción



3.7.1 Algoritmo de Polling (Interrogación)

Supongamos que el host desea escribir salidas a un controlador vía una puerta.

Para esto se ejecuta el siguiente loop:

- El host lee el *bit busy* para ver si está desocupado.
- El host pone en 1 el bit *write* del registro de comando y escribe un byte en el registro data-out.
- El host pone en 1 el bit *command ready* en 1 del registro de comando.
- El controlador lee el registro de comando, lee el registro data-out y lo pasa al dispositivo.
- El controlador limpia el bit *comand ready* y el *bit busy* para indicar que terminó.

El Algoritmo de Polling es eficiente, pero puede llegar a no serlo cuando se ocupa repetidamente.

Un esquema distinto es permitir al dispositivo notificar a la CPU cuando está listo. Este mecanismo se denomina interrupción.

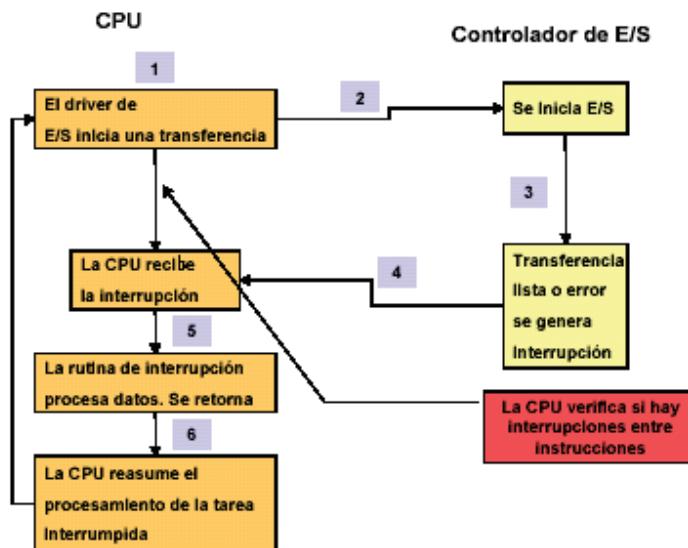
3.7.2 Interrupciones

El hardware de la CPU tiene una línea llamada línea de requerimiento de interrupción. Después de ejecutar cada instrucción, la CPU lee esta línea. Si se detecta que está alta, la CPU salva el estado y salta a una rutina de manejo de interrupciones en una dirección fija de memoria.

Esta rutina determina la causa de la interrupción, realiza el procesamiento necesario y vuelve a ejecutar la instrucción siguiente.

Muchas arquitecturas tienen un vector de interrupción que evita la rutina de manejo direccionando la rutina especializada.

Gráfica 68. Manejo de interrupciones



3.7.3 Acceso Directo a la Memoria (DMA)

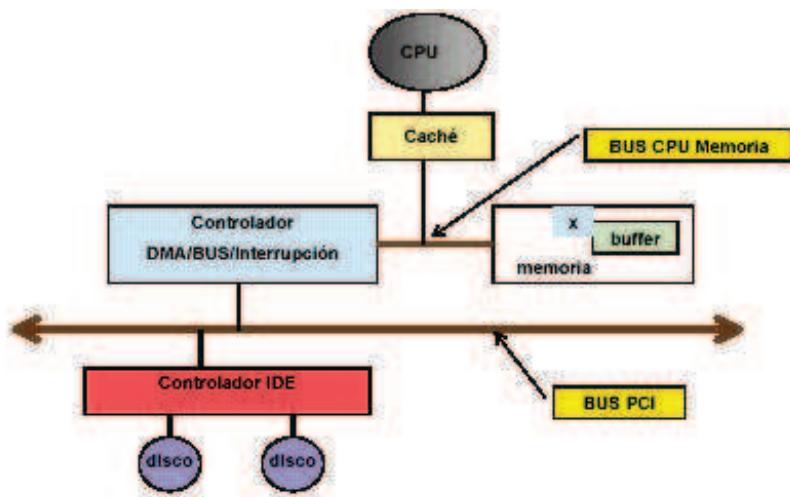
Los métodos de Polling e Interrupciones se basan en programas que controlan la E/S. Estos métodos se denominan PIO (Programmed I/O).

Muchos computadores evitan sobrecargar laCPU con PIO incorporando un procesador especializado cuando se realizan grandes transferencias de datos, por ejemplo en discos. El procesador especializado se denomina **DMA**.

Para iniciar una transferencia, la CPU sólo escribe un comando **DMA** en la memoria.

El comando contiene: puntero al comienzo del bloque fuente, un puntero al bloque destino y el número de bytes a transferir.

Gráfica 69. Hardware DMA



Etapas en una transferencia DMA

1. El driver se informa que debe transferir datos del disco a la dirección X.
2. El driver informa al controlador del disco que debe transferir C Bytes del disco al buffer en X.
3. El controlador de disco inicia transferencia DMA.
4. El controlador del disco envía cada byte al controlador de DMA.
5. El Controlador DMA transfiere bytes a X incrementando la dirección de memoria y decrementando C en cada paso hasta que C==0.
6. Cuando C==0 el DMA interrumpe la CPU informando que la transferencia está completa.

Actividad final:

Revise la lista realizada al inicio del capítulo y corrija, adicione o complete si es necesario.

CAPÍTULO 4. ADMINISTRACIÓN DE ARCHIVOS²⁷

Actividad inicial:

Usted conoce cómo es la organización interna del sistema operativo para manejar la estructura de archivos y directorios?

Qué tipos de estructuras de archivos y directorios conoce. De cuáles sistemas operativos?

4.1 El sistema de archivos

El sistema de archivos es la parte más visible de un sistema operativo.

Un sistema de archivos está formado por:

- Una colección de archivos.
- Una estructura de directorios.

4.1.1 El archivo

El S.O proporciona una visión lógica uniforme de la información almacenada.

Un archivo es una unidad lógica de almacenamiento. Los archivos son mapeados a través el S.O en dispositivos físicos.

Desde la perspectiva del usuario, no es posible escribir datos en el almacenamiento secundario si no es a través de un archivo.

Un archivo tiene una estructura que está definida por su tipo

4.1.2 Atributos de archivos

Los siguientes son atributos de un archivo:

- **Nombre.** Nombre simbólico.
- **Tipo.** Información necesaria para sistemas que soportan tipos diferentes.
- **Localización.** Algun puntero a un dispositivo y lugar dentro del dispositivo donde se almacena el archivo.

²⁷ Todas las gráficas de este capítulo fueron extraídas del documento en formato pdf: Sistemas Operativos. Profesor Javier Cañas. Capítulos 9: Sistema de archivos y Capítulo 10: Estructura de archivos.

- **Tamaño.** Tamaño actual en bytes o palabras o bloques.
- **Protección.** Información de control de acceso. Establece quién puede leer, escribir y ejecutar.
- **Tiempo, fecha, identificación de usuario.** Para protección, seguridad y monitoreo de uso.

4.1.3 Operaciones sobre archivos

Un archivo es un tipo abstracto de datos. Para definirlo, se deben considerar las operaciones que se pueden realizar sobre él. Estas operaciones están definidas a través de **llamadas al sistema**. Las operaciones básicas sobre archivos son:

- **Crear un archivo.** Para esto se requieren dos acciones; crear espacio en el sistema de archivos y ubicarlo en un directorio.
- **Escribir**
- **Leer**
- **Reposicionar (seek)**
- **Eliminar**
- **Truncar**

Operaciones de mayor complejidad son posibles con estas operaciones primitivas.

Por ejemplo copiar = crear + leer + escribir.

¿Qué significa abrir un archivo?

La mayoría de las operaciones sobre archivos involucran una navegación constante sobre el directorio buscando un archivo con un determinado nombre.

Abrir un archivo significa poner en una tabla de acceso rápido información de los archivos usados por un proceso. De esta forma se mantiene en la tabla un índice de acceso rápido.

Estas operaciones se denominan **open y close**:

- **Open.** Toma un nombre de archivo y busca en el directorio. Retorna un índice a la tabla de archivos.
- **Close.** Libera una entrada en la tabla de archivos.

4.1.4 Tipos de archivos

Una decisión de diseño de un sistema de archivos y en general de un S.O es la facilidad de reconocer y soportar distintos tipos de archivos.

Por ejemplo si se imprime un archivo binario, se genera basura. Lo razonable es prevenir que esto ocurra.

La forma más común es incorporar en el nombre el tipo del archivo, es decir su **extensión**.

Una forma es la usada en MS-DOS con el formato punto (ejemplo: tarea.com, carta.txt, etc.)

En Unix cada archivo tiene al comienzo un número mágico que indica el formato del archivo.

Ejemplos de tipos de archivos

Tipo de Archivo	Extensión usual
Ejecutables	exe, com, bin,
Objeto	obj, o
Código fuente	c, cpp, pas, asm, a, p, f77
Batch	bat, sh
Texto	txt, doc
Procesador de texto	wp, tex, rtf, doc
Bibliotecas	lib, a
Impresión o visualización	ps, dvi, gif
Grupos de archivos	arc, zip, tar, arj

4.2 Métodos de acceso

Los archivos almacenan información. Hay muchas maneras de accesar a la información contenida en un archivo.

Existen sistemas que proporcionan sólo una forma y otros múltiples formas de acceso.

Las formas básicas de acceso son dos:

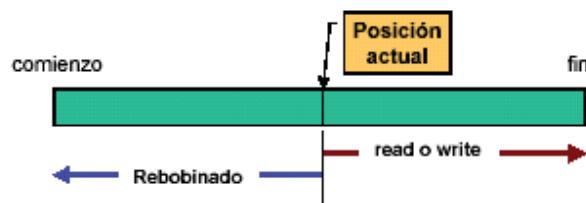
- Acceso secuencial.
- Acceso directo.
- Acceso indexado

4.2.1 Acceso secuencial

En este tipo de acceso la información es procesada en orden y la mayoría de operaciones son lecturas y escrituras.

Este método está basado en el modelo de cinta magnética.

Gráfica 70. Acceso secuencial



4.2.2 Acceso directo

También se denomina acceso relativo. Un archivo se construye con base a **registros lógicos** de tamaño fijo. Se pueden leer en cualquier orden.

Se basa en el modelo de disco. Cada registro tiene un número (No. de bloque ó No. de registro). Los bloques se numeran consecutivamente: 0, 1, ...

Si el tamaño de un bloque es L, solicitar el bloque N significa buscarlo en la posición $L \times N$ dentro del archivo.

Gráfica 71. Acceso directo

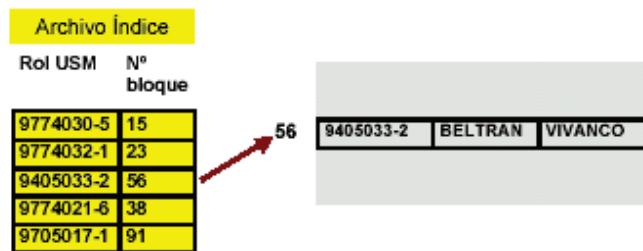


4.2.3 Acceso indexado

Se construye sobre el método de acceso directo.

Un archivo llamado archivo índice contiene punteros a varios bloques. Encontrar una entrada a un archivo involucra buscar primero en el índice.

Gráfica 72. Acceso indexado



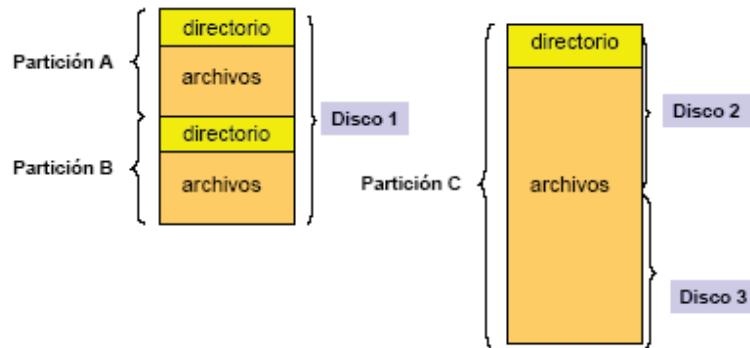
4.3 Directorios

¿Cuántos archivos se mantienen en cualquier sistema?

Para manejar tanta información, es necesario organizarla. Esta organización requiere dos niveles:

- **Particiones** (volúmenes o minidiscos).
- **Directorios** (tabla de contenido de volúmenes).

**Gráfica 73. Ejemplo de organización
Ejemplo de Organización**



4.3.1 Operaciones sobre un directorio

Un directorio puede organizarse de varias maneras. Las operaciones que se pueden hacer sobre un directorio son:

- **Búsqueda de archivos**
- **Crear archivos**
- **Borrar archivos**
- **Ver directorios (nombres de archivos)**
- **Renombrar un archivo**
- **Navegar por un sistema de archivos**

4.3.2 Estructuras de directorios

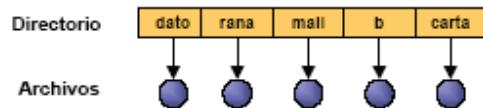
Los directorios se pueden organizar de diversas maneras. Las distintas posibilidades son:

- **Un nivel**
- **Dos niveles**
- **Árboles**
- **Grafos acíclicos**

Gráfica 74. Directorio de un nivel

Directorio de un Nivel

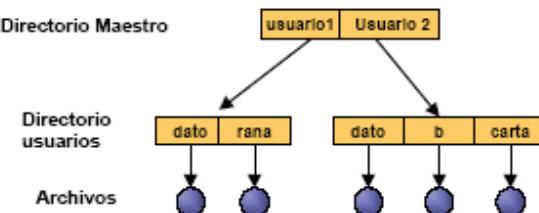
- Todos los archivos están contenidos en un solo directorio
- Los nombres de los archivos deben ser únicos
- La diferenciación entre los tipos sólo se da por el nombre



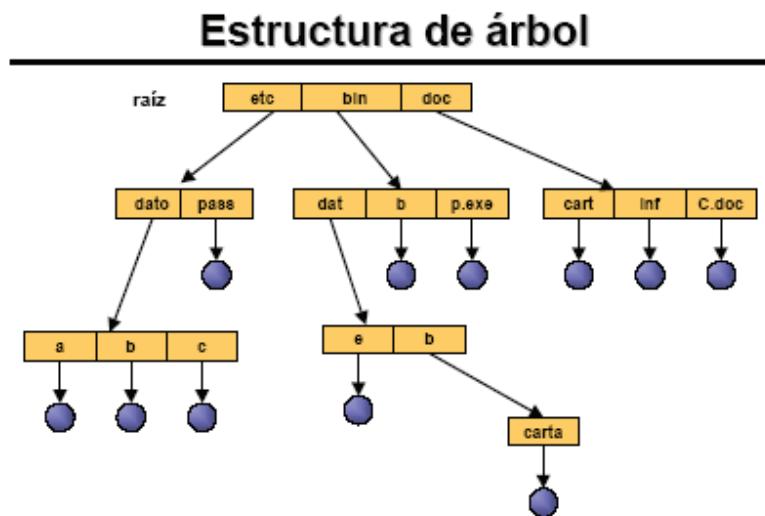
Gráfica 75. Directorio de dos niveles

Directorio de dos Niveles

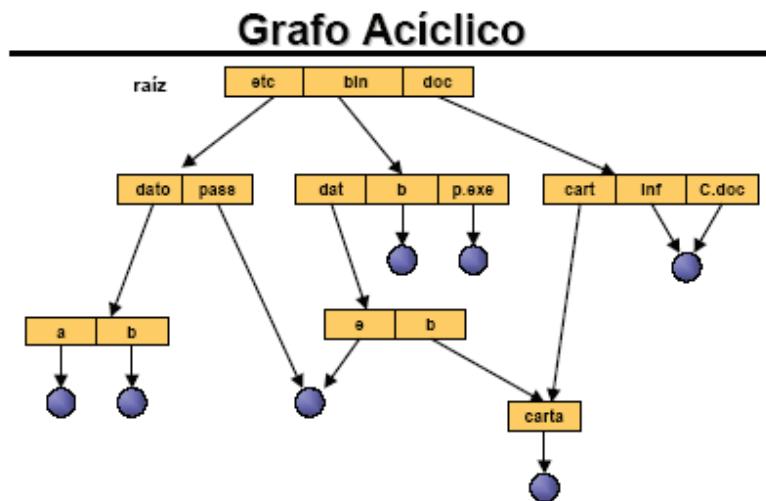
- Un directorio de un nivel genera confusión de nombres entre usuarios distintos
- En un Directorio de dos niveles, cada usuario tiene su propio directorio



Gráfica 76. Directorio de árbol



Gráfica 77. Grafo cíclico



4.4 Protección

Cuando la información se almacena en un sistema computacional, se deben tomar dos tipos de protección:

- Daños físicos (confiabilidad).
- Accesos no autorizados (protección).

La **confiabilidad** se logra por la mantención de respaldos y duplicación, mientras que la **protección** se logra por varias vías:

- Removiendo discos, usando llaves
- Protección por software

4.4.1 Tipos de acceso

Los extremos de la protección son:

- No permitir el acceso a nadie más.
- Permitir libre acceso a todos.

Entre estos dos extremos está el **control de acceso**. Las operaciones que normalmente se controlan son:

Lectura	Escritura	Ejecución
Agregar al final	Borrar	Ejecución

4.4.2 Listas de acceso

Un esquema general es asociar a cada archivo y directorio una **lista de acceso**. La lista de acceso especifica el nombre del usuario y los tipos de accesos que son permitidos.

Al hacer un acceso se verifica la lista. Si no hay acceso ocurre una violación y se niega el permiso. La dificultad de la lista de acceso es su gran tamaño.

Una implementación práctica de listas de acceso, es a través de la definición de grupos de usuario, con diferentes niveles de acceso.

Muchos sistemas operativos reconocen tres tipos de usuarios:

- **Dueño.** El usuario que creo el archivo. Conocido como **propietario**.
- **Grupo.** Conjunto de usuarios que comparten el archivo.
- **Universo.** Todos los usuarios del sistema.

Un ejemplo es UNIX/LINUX que usa tres bits para cada tipo de usuario: `rwx`

r = acceso de lectura. Valor de 4. $2^2=4$

w = acceso de escritura. Valor de 2. $2^1=2$

x = acceso de ejecución. Valor de 1. $2^0=1$

```
-rw-rw-r- 1jcr academ 32400 sep 4 08:30 archivos.ps
```

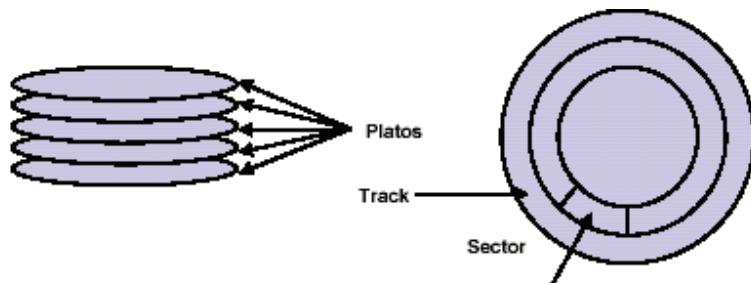
4.5 Estructura de archivos

Trata sobre aspectos relacionados con el almacenamiento y acceso de archivos almacenados en discos. El sistema de archivos se almacena en disco.

Para mejorar el desempeño de la E/S de un sistema, las transferencias entre discos y memorias se realizan en bloques.

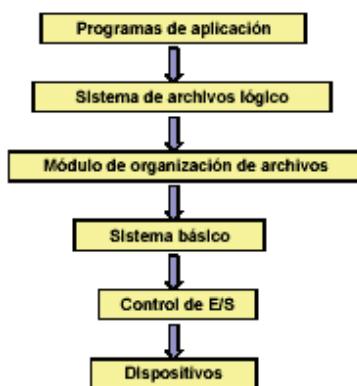
Cada bloque está formado de uno o más sectores (32B a 4096B).

Gráfica 78. Organización de un disco



- Números típicos (dependientes del tamaño del disco):
 - 500 a 2,000 tracks por superficie
 - 32 a 128 sectores por track
 - Un sector es la unidad más chica que se puede leer o escribir
- Tradicionalmente todos los Tracks tienen el mismo número de Sectores:
 - Densidad constante de bits: graba más sectores en los tracks de afuera

Gráfica 79. Organización de un sistema de archivos



- **Control de E/S.** Drivers de discos y manejador de interrupciones.

- **Sistema básico.** Se ocupa de enviar comandos genéricos a los drivers apropiados para leer y escribir bloques físicos en el disco.
- **Módulo de organización de archivos.** Se encarga de convertir direcciones lógicas de bloques en direcciones físicas para el sistema básico. También forma parte de este módulo el manejador de espacio libre.
- **Sistema de archivos lógico:** Usa la estructura de directorios para proporcionar al módulo de organización de archivos la información que necesita a partir de un nombre simbólico.
- **Programas de aplicación.**

¿Cómo se crea un nuevo archivo?

Para crear un nuevo archivo:

- El programa de aplicación llama al sistema de archivos lógico.
- El sistema de archivos lógico lee el directorio apropiado en memoria, lo actualiza con la nueva entrada y lo escribe en disco.
- Una vez creado se puede usar para E/S.
- Para evitar recorrer el directorio en E/S, el archivo se abre.
- Abrir significa copiar sus parámetros en la tabla de archivo.

Gráfica 80. Estructura de la tabla de archivos

Índice	Nombre	Permisos	Fecha Acceso	Puntero A disco
0	rana.cpp	rw-rw-rw-		
1	mail.txt	rw		
2				
n				

4.5.1 Sistema de archivo

El aspecto clave de la implementación del almacenamiento de archivos es el registro de los bloques asociados a cada archivo. Una vez más, cada sistema de archivos implementa métodos distintos para solucionar este problema.

Un **bloque** está compuesto por un determinado número de sectores que se asocian a un único archivo. Un archivo, por tanto, se almacena en uno o más bloques de sectores.

Un aspecto muy importante es la elección del tamaño del bloque, para esto hay que entender que si el tamaño del bloque es muy grande, aun cuando el archivo sea de un tamaño muy pequeño, se asignará el bloque entero como lo que se desperdiciará gran parte de la capacidad del disco.

Por otra parte, si el tamaño del bloque es demasiado pequeño para almacenar un archivo, harán falta muchos bloques con lo que se producirá un retraso en la lectura del archivo al tener que localizar el disco todos los bloques que componen dicho archivo. Una vez más, se ha de llegar a la solución de compromiso, eligiendo un tamaño de bloque lo suficientemente pequeño para no desperdiciar capacidad de disco pero lo suficientemente grande como para no ralentizar en exceso la lectura de los archivos. Diversos estudios realizados indican que el tamaño medio de los archivos en sistemas UNIX y MS-DOS ronda el 1 Kbyte, así pues, son adecuados los tamaños de bloque de 512 Bytes, 1 Kbyte o 2 kybytes.

Si se elige un tamaño de bloque de, por ejemplo, 2 kbytes en un disco cuyo sector tiene 512 bytes, cada bloque estará compuesto por cuatro sectores.

Para manejar los bloques asociados a cada archivo, se pueden utilizar varias técnicas, las cuales se tratan más adelante.

4.6 Métodos de asignación

El acceso directo proporciona gran flexibilidad en la implantación de archivos. El principal problema es cómo asignar el espacio a los archivos de forma tal que el disco sea utilizado eficientemente y que los accesos sean rápidos.

Los tres principales métodos de asignación son:

- **Contigua**
- **Enlazada**
- **Indexada**

4.6.1 Asignación contigua

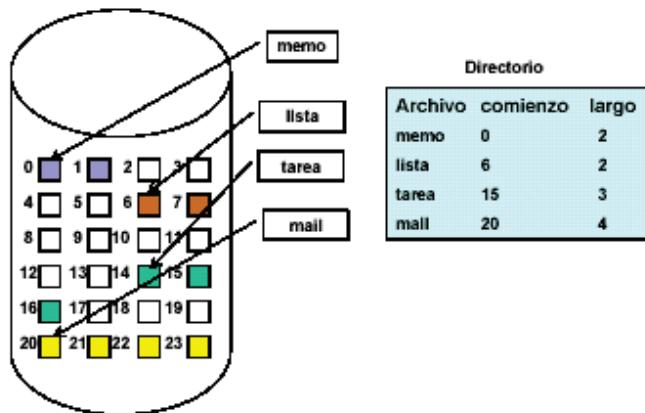
Se requiere que cada archivo ocupe un conjunto contiguo de bloques. Las direcciones definen un orden lineal.

Con esta organización se minimiza el movimiento del cabezal al limitarlo sólo a un track al pasar de un cilindro a otro. Este sistema provee un buen desempeño.

Una entrada al directorio sólo necesita la dirección inicial y el largo del bloque.

La asignación contigua genera ***fragmentación externa***.

Gráfica 81. Asignación contigua de espacio



Además de la fragmentación se genera otro problema: ¿Cuando se crea un archivo, cómo se sabe el tamaño del archivo final?.

Si se asigna poco espacio, el archivo no se puede extender, si se asigna mucho espacio, se pierde por fragmentación interna.

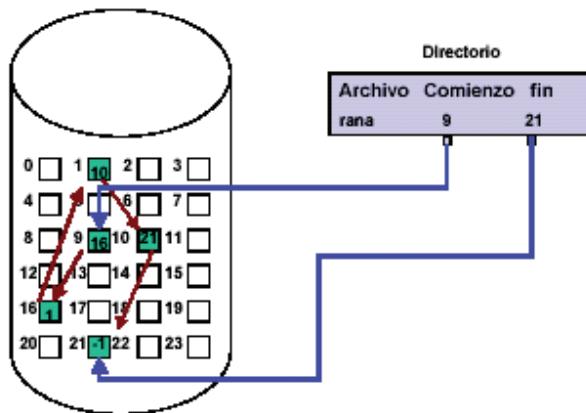
4.6.2 Asignación enlazada

Cada archivo es una ***lista enlazada de bloques de disco***.

El directorio contiene un puntero al primer y último bloque. Cada bloque contiene un puntero al siguiente bloque. Este puntero está oculto al usuario.

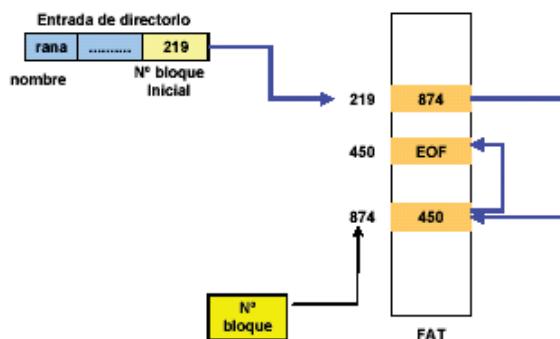
No hay fragmentación externa con este método. Tampoco es necesario definir el tamaño del archivo cuando es creado. El archivo sigue creciendo mientras haya bloques libres.

El gran problema es que sólo es eficiente para archivos de acceso secuencial. Otra desventaja es el espacio requerido para punteros.

Gráfica 82. Asignación de espacio enlazada**FAT: File Allocation Table**

Una variación importante de la asignación enlazada es el uso de FAT (MS-DOS y OS/2). Una sección del disco al comienzo de cada partición contiene una tabla. Esta tabla contiene una entrada para cada bloque de disco y su índice por número de bloque.

Una entrada al directorio contiene el número de bloque del primer número de bloque del archivo. Se puede observar que este esquema puede resultar en muchos movimientos del cabezal lector, al menos que la FAT sea capturada en una caché.

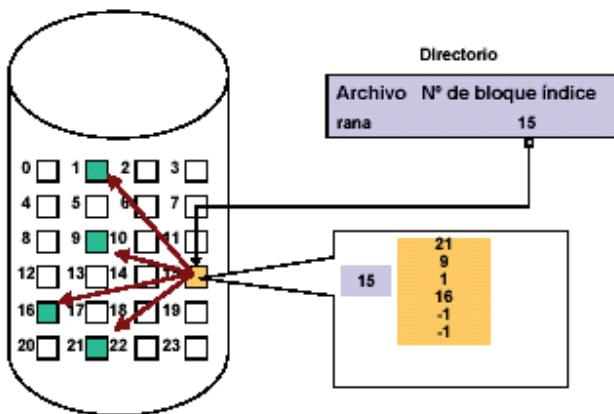
Gráfica 83. FAT**4.6.3 Asignación indexada**

La asignación enlazada resuelve la fragmentación externa y el problema del tamaño inicial del archivo.

Si no se usa FAT, la asignación enlazada no puede soportar acceso directo ya que los punteros a bloques están dentro de los mismos bloques. Este problema se resuelve poniendo junto todos los punteros a bloques en un mismo lugar.

Cada archivo tiene su propio índice de bloques formado por un arreglo de direcciones de bloques. Cuando un archivo es creado, cada entrada tiene el valor NIL(nulo).

Gráfica 84. Asignación de espacio indexada



Soporta acceso directo sin ser afectada por fragmentación externa
Se gasta espacio. El overhead que impone el bloque de punteros es superior al espacio perdido en punteros en una asignación enlazada.
El sistema operativo UNIX tiene un esquema indexado en el cual la información de un archivo se almacena en una estructura llamada *inode*.

Esta técnica es llamada **asignación mediante una lista ligada y un índice**, que intenta eliminar los defectos de la anterior. En esta técnica se crea una tabla con un registro por cada uno de los bloques del disco, en cada registro se indica si dicho bloque está libre (null) o cuál es la dirección del siguiente bloque (en caso de que ese bloque pertenezca a un determinado archivo). De esta forma, en el directorio se asocia con el nombre del archivo el número del bloque en el que comienza dicho archivo; con este dato y, mediante la tabla, se puede averiguar la dirección de todos los bloques que componen dichos archivos simplemente siguiendo la lista ligada.

Con esta organización, todo el bloque estará disponible para los datos. Además, el acceso a un determinado bloque es mucho más rápido, ya que aunque también haya que seguir la cadena de bloques como en la asignación en forma de lista ligada, al estar la tabla en memoria, estas consultas son mucho más rápidas y no es necesario acceder al disco.

Número de Bloque	Dirección siguiente bloque
0	
1	13
2	3
3	7
4	0
5	
6	2
7	1
8	
9	
10	11
11	n-2
12	
13	0
.	
.	
n-2	4
n-1	

Tabla de registros de asignación continua de bloques

La desventaja que tiene este método es que toda la tabla de registros deberá estar en la memoria principal permanentemente, con lo que la memoria consumida para almacenar la tabla no estará disponible para ser usada por otros procesos. Esto llega a ser un gran problema en el supuesto de discos con un gran número de bloques, ya que en la tabla de registros puede llegar ocupar gran parte de la memoria principal del ordenador o incluso desbordarla.

Esta es la técnica utilizada por el MS-DOS y por Windows. En este caso a la tabla de registros se denomina **FAT (File Allocation Table)** y se puede encontrar en sus dos versiones: **FAT16** y **FAT32**, dependiendo de si los bloques se direccionan con 16 o con 32 bits respectivamente.

Ejemplo: Ejemplo de funcionamiento de lectura de un fichero en MS-DOS.

Supongamos que un usuario solicita leer el fichero prueba.txt. En este caso, el sistema operativo leerá el directorio activo (y que se trata de una ruta relativa) en busca de la entrada correspondiente a dicho archivo. Si éste existe, se hallará un registro con cierta información relativa a dicho archivo (como son los atributos del archivo y, también, el bloque del disco en el que el archivo comienza). Con dicha información busca en la FAT, que se encuentra en la memoria principal, el registro perteneciente a ese bloque y en él se encontrará la dirección del siguiente bloque en el que el archivo está escrito. Repitiendo esta operación hasta en la dirección del siguiente bloque sea 0 obtenemos la lista completa de bloques en los que el archivo está almacenado.

En la tabla, se puede ver que el archivo que comienza en el bloque No. 6, continuará en los bloques: 2, 3, 7, 1 y 13.

El lector también podrá ver que los bloques 0, 5, 8, 9, 12 y n-1 están libres y hay otro fichero almacenado en los bloques 10, 11, n-2 y 4.

Por último, los sistemas operativos como UNIX y Linux utilizan un sistema de archivos basados en **i-nodos**. En esta técnica se asocia a cada archivo una pequeña tabla, llamada i-nodo, que contiene los atributos y direcciones en disco de los bloques del archivo.

Atributos
Dirección del bloque 1
Dirección del bloque 2
Dirección del bloque 3
Dirección del bloque 4
.
.
.
Dirección del bloque n

I-nodo de UNIX

Las últimas entradas del i-nodo se reservan para cuando el archivo ocupa más bloques de los que el i-nodo es capaz de almacenar y pueden contener la dirección de otro bloque en el que se guardan las demás direcciones de los bloques del archivo. A este bloque se le llama **bloque indirecto**. En este caso de que con este bloque extra no haya suficiente espacio para guardar todas las direcciones de los bloques del archivo, existe la posibilidad de utilizar un bloque doblemente indirecto e, incluso, un tercer bloque triplemente indirecto.

Cuando UNIX abre un archivo, lo primero que hace es cargar en memoria su i-nodo correspondiente para que el proceso sea lo más rápido posible.

4.7 Tipos de sistemas de archivo

Como se dijo anteriormente, existen distintos tipos de sistemas de archivos, siendo los siguientes los más utilizados para el entorno monousuario y multiusuario.

- **FAT16.** Se puede acceder a este sistema de archivos desde MS-DOS, Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP y Windows Server 2003. Permite trabajar con particiones de hasta 2 GB, las unidades de asignación de 32 KB, el tamaño máximo de un archivo es de 2 GB, los volúmenes pueden llegar hasta 2 GB, no distingue entre mayúsculas y minúsculas en los nombres de archivos y no soporta dominios.
- **FAT32.** Se puede acceder a este sistema de archivos desde Windows 95 OSR2, Windows 98, Windows 2000, Windows XP y Windows Server 2003.

Permite trabajar con particiones mayores de 2 GB, las unidades de asignación son de 4 KB, el tamaño máximo de un archivo es de 4 GB, los volúmenes pueden llegar hasta 2TB (en Windows 200/XP/2003 solo hasta 32 GB), no distingue entre mayúsculas y minúsculas en los nombres de archivo y no soporta dominios.

- **NTFS 4 (NT File System 4).** Es el sistema desarrollado para Windows NT 4 que permite nombres de archivo de hasta doscientos cincuenta y seis caracteres, ordenación de directorios, atributos de acceso a archivos, reparto de unidades en varios discos duros, reflexión de discos duros y registro de actividades, distingue entre mayúsculas y minúsculas en los nombres de archivo y soporta dominios de Windows NT.
- **NTFS 5 (NT File System 5).** En Windows 2000 Server se incluyeron mejoras que permitirán utilizar el Directorio Activo, dominios de Windows 2000/2003, cuotas en disco para cada usuario, cifrado y comprensión de archivos, almacenamiento remoto, una herramienta de desfragmentación y utilización de enlaces de archivos similares a los realizados en UNIX. Sus volúmenes pueden llegar hasta 16 TB menos 64 KB y el tamaño máximo de un archivo sólo está limitado por el tamaño del volumen. Distingue entre mayúsculas y minúsculas en los nombres de archivos.
- **Sistema de Ficheros Extendidos 2 (ext2fs).** Es uno de los más eficientes y flexibles sistemas de archivos. Se puede acceder desde UNIX/LINUX, permite hasta 256 caracteres en los nombres de los ficheros, el tamaño del máximo del sistema de archivos es de 4 Terabytes y el tamaño máximo de un archivo es de 2GB. Distingue entre mayúsculas y minúsculas en los nombres de archivo.
- **Servicios de almacenamiento de Novell (NSS).** Es el más reciente sistema de archivos propietarios de Novell que se ejecuta desde NetWare 5 y NetWare 6. Permite hasta 8 billones de archivos por servidor pudiendo haber un millón de archivos abiertos simultáneamente, el tamaño del bloque se adapta automáticamente en función del tamaño del volumen, el volumen máximo de un archivo y de un volumen es de 8 TB, y puede haber hasta 255 volúmenes montados en un servidor.
- **HPFS (High Performance File System).** Se puede acceder desde OS/2 y se creó para resolver todos los problemas del sistema de archivos FAT. Utilizan un tamaño de sector de 512 bytes y pueden tener un tamaño máximo de 2 GB.

4.8 Mecanismos de protección de los ficheros

Dentro de los mecanismos de protección de la información contenida en los ficheros se encuentran los siguientes:

- Las unidades RAID
- Los permisos de acceso
- Las copias de seguridad

4.8.1 Las unidades Raid

El término **RAID** significa **Redundant Array of Independent Disk (Matriz Redundante de Discos Independientes)**

La filosofía de esta tecnología consiste en disponer de varias unidades de disco, conectadas entre sí, por medio de controladoras, software o combinación de ambas; de manera que cuando una unidad física de disco falle o se venga abajo, los datos que se encontrarán en dicha unidad no se pierdan sino que se reconstruyan usando la paridad de los mismos (el sistema operativo ve a la matriz como si ésta fuese una sola).

Las configuraciones definidas en RAID son las siguientes:

RAID 0. La información se divide entre los discos del sistema, de forma que no se establece ningún tipo de redundancia.

- **Ventajas:** Proporciona alto rendimiento (tiempos de acceso muy bajos, posibilidad de acceso en paralelo). No tiene costo adicional. Se emplea toda la capacidad del disco.
- **Inconvenientes:** No es verdaderamente un disco RAID ya que no presenta integridad de los datos. Un error en uno de los discos implica la pérdida total de los datos.

RAID 1. También conocido como **MDA (Mirrored Disk Array)**, en esta configuración los discos se asocian por parejas y cada una de ellas almacenará la misma información. Cada pareja está formada por un disco primario, donde se leen y se escriben los datos, y un disco espejo, donde solamente se escriben las modificaciones y del que se leerán datos cuando el primario falle.

- **Ventajas:** En caso de error de uno de los disco se recuperan todos los datos. Es la arquitectura más rápida que presenta tolerancia a fallos. Con un mínimo de dos discos es suficiente.

- **Inconvenientes:** Es bastante caro, ya que se emplea el doble espacio del necesario.

RAID 2. Emplea múltiples discos, como en el nivel RAID 0, pero algunos de estos son empleados para guardar también los códigos de control de error. Este nivel tiene un costo bastante elevado ya que se necesitan muchos discos para mantener los códigos de error.

Gracias a cómo están distribuidos los datos en los discos, se consigue mejorar la velocidad de transferencia principalmente en la lectura, ya que es posible emplear todos los discos en paralelo.

Estos discos, aunque proporcionan un buen rendimiento, no son muy empleados, ya que los niveles 1,3 y 5 proporcionan una mejor relación costo/rendimiento.

- **Ventajas:** Se consigue aumentar la velocidad de transferencia. Es posible recuperar los datos a partir de los códigos redundantes de error.
- **Inconvenientes:** Es una solución cara ya que se requiere mucho espacio para almacenar los códigos de error. El tiempo de lectura de los datos es bastante lento, aunque los datos se separen en los diferentes discos.

RAID 3. Es un sistema de discos en paralelo con discos de paridad para la corrección de errores. También conocida como PDA (Parallel Disk Array). Al igual que RAID 0 y RAID 2, almacena la información en varios (dividida a nivel de byte) pero se deja uno de ellos para almacenar los dígitos de paridad generados a partir de dichos datos. La información de paridad se utiliza para comprobar la consistencia de los datos almacenados.

Este nivel RAID es una buena alternativa para aplicaciones de velocidad de transferencia alta, ya que gracias a la distribución de datos se pueden emplear todos los discos en paralelo.

- **Ventajas:** Se consigue un alto rendimiento para aplicaciones que necesiten velocidades de transferencia alta. Es posible recuperar datos gracias al disco de paridad.
- **Inconvenientes:** Si se daña el disco de paridad y se pierde toda la información redundante el tiempo de escritura es bastante alto.

RAID 4. Es un sistema de discos independientes con disco de control de errores. También conocido como **IDA (Independet Disk Array)**. Es parecido al RAID 3 pero con la diferencia de que los datos se dividen en bloques o sectores. Los bloques de datos que se distribuyen en los diferentes discos, al ser más grandes, hace que se consiga un rendimiento superior en la escritura.

- **Ventajas:** Mantiene la integridad de los datos. Buen rendimiento en la escritura de datos.
- **Inconvenientes:** Si se pierde un disco de paridad se pierde toda la información redundante. Menor rendimiento en lectura de datos.

RAID 5. Es un sistema de discos independientes con integración de códigos de error, mediante paridad. En RAID 5 los datos y la paridad se guardan en los mismos discos, por lo que se consigue aumentar la velocidad de demanda. La diferencia respecto al RAID 3 es que en RAID 5 se guarda la paridad de los datos dentro del disco y no hace falta un disco para guardar dichas paridades.

La paridad se genera haciendo un XOR de los datos A0, B0, C0, D0 y creando la zona de paridad PAR0; como se ve la paridad nunca se guarda en los discos que contienen los datos que han generado dicha paridad, ya que en el caso de que uno de ellos se estropeará (como por ejemplo el dato A0) bastaría con regenerar las bandas B0, C0, D0 y Par0 para que el dato volviera a reestablecerse.

- **Ventajas:** Alto rendimiento en aplicaciones con gran demanda de velocidad. No se desaprovecha ningún disco exclusivamente para almacenar códigos de paridad. Se pueden recuperar los datos.
- **Inconvenientes:** Bajo rendimiento en escritura. Se requiere un mínimo de tres discos.

RAID 6. Es un sistema de discos independientes con integración de códigos de error mediante paridad doble. Es igual que RAID 5, con la diferencia de que se guardan dos paridades para cada bloque de información, cada una de ellas alojada en un disco diferente (también diferente al de los datos).

- **Ventajas:** Es posible recuperar varios errores al mismo tiempo. El nivel de integridad de datos es muy elevado.
- **Inconvenientes:** El rendimiento en escritura de datos es muy bajo.

RAID 7. Es igual a RAID 4, aunque utiliza un sistema operativo en tiempo real residente en el equipo, que es el que gestiona los accesos a los discos.

RAID 10. Es igual a RAID 1, salvo que se incluyen otros discos donde se almacena otra copia completa de la información. Se utiliza para conseguir un mayor tiempo de acceso.

RAID 53. Es igual que RAID 3, salvo que se incluyen otros discos donde se almacena otra copia completa de la información. Se utiliza para conseguir un mayor tiempo de acceso.

RAID 0 + 1. Es una combinación de las técnicas RAID 0 y RAID 1, de forma que la información se duplica en conjuntos de discos, los cuales almacenan los datos distribuidos con ellos.

4.9 Administración del espacio libre

Como el espacio de almacenamiento es limitado, es necesario reutilizar el espacio de archivos que se borran para dejar espacio a los nuevos archivos que se crean. Para registrar el espacio libre, el sistema operativo mantiene una lista de espacio libre. Esta lista registra todos los bloques que están libres.

Métodos para implementar la lista de bloques libres:

- Bit vector
- Listas enlazadas
- Agrupación
- Conteo

4.9.1 El método Bit vector

También se denomina bit map. Cada bloque se representa por un bit. Si el bloque está libre se representa por 1 y si está ocupado por 0.

Ejemplo:

Bloques libres: 2, 3, 4 ,5, 8, 9, 10, 11, 12, 13, 17, 18, 25,....

Bit vector: 001111001111100011000001,.....

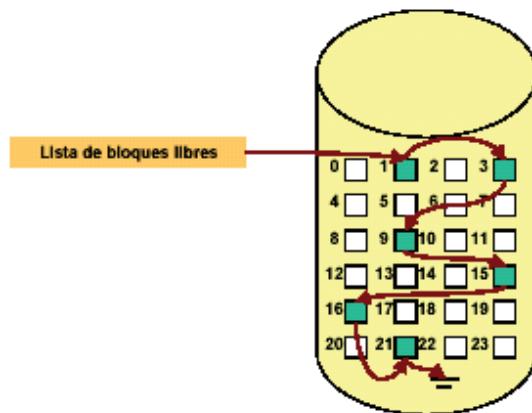
Ventaja: simple.

Desventaja: para que sea eficiente hay que mantener la estructura completa en memoria principal. En la práctica sólo es posible para discos chicos.

4.9.2 El método lista enlazada

Otra alternativa es enlazar todos los bloques libres de disco, almacenando un puntero al primer bloque libre en un lugar especial de disco.

Este esquema no es eficiente ya que para recorrer la lista se requiere mucho tiempo de E/S (no es muy frecuente hacer esto).

Gráfica 85. Lista enlazada de bloques libres

4.9.3 El método de agrupación

Una modificación a la lista enlazada de bloques es almacenar la dirección de n bloques libres en el primer bloque libre. Las primeras n-1 direcciones de bloques corresponden a bloques libres y la última dirección contiene la dirección de otros n bloques y así...

El S.O.UNIX (AT&T) tiene esta implementación.

4.9.4 El método de conteo

Este método aprovecha el hecho que normalmente muchos bloques contiguos se asignan o liberan simultáneamente, especialmente en algoritmos de asignación contigua de memoria.

En vez de almacenar la dirección de n bloques, sólo se almacena la dirección del primer bloque libre y el número de los n bloques contiguos que le siguen.

De esta manera, cada entrada de la lista está formada por una dirección de bloque y una cuenta.

Actividad final:

Ahora que ya conoció o repasó acerca de la organización interna del sistema operativo para manejar la estructura de archivos y directorios puede realizar un pequeño mapa conceptual en donde resuma los tipos de estructuras de archivos y qué sistemas operativos los manejan. Este material le servirá para la preparación de la evaluación final del curso?

CAPÍTULO 5. PROTECCIÓN Y SEGURIDAD²⁸

Actividad inicial:

Imagine cómo será el sistema de seguridad y protección de un sistema operativo. Además imagine qué tiene que proteger el sistema operativo y contra quién o quélo protegen.

De acuerdo a esto, elabore un ensayo de 2 páginas en donde desarrolle sus hipótesis acerca del tema.

5.1 Protección

Los mecanismos de protección proveen un control de acceso limitando el tipo de accesos que se pueden hacer a los archivos.

La protección considera los mecanismos para asegurar que otros recursos como memoria, CPU y dispositivos, sean operados sólo por usuarios con autorización del sistema operativo.

5.2 Objetivos de la protección

Cada proceso se debe proteger de las demás actividades del S.O. La protección se refiere a mecanismos para el control de acceso de programas.

¿Para qué proteger?

- Prevenir violaciones de acceso intencionales.
- Asegurar que cada proceso use recursos sólo en forma consistente con las políticas de uso.
- Asegurar confiabilidad detectando errores latentes entre interfaces.
- Distinguir usos autorizados y no autorizados de recursos.

5.2.1 Políticas de protección

La protección sirve para generar un mecanismo que fuerce políticas de uso de recursos.

Las políticas pueden ser:

- Fijas e incorporadas en el diseño del S.O.

²⁸ Todas las gráficas de este capítulo fueron extraídas del documento en formato pdf: Sistemas Operativos. Profesor Javier Cañas. Capítulos 12: Protección y Capítulo 13: Seguridad.

- Definidas por usuarios individuales para proteger archivos y programas.

Hay que separar dos conceptos: mecanismos y políticas:

- **Mecanismo.** Determina cómo algo debe ser hecho.
- **Política:** Decide qué debe ser hecho.

5.3 Dominios de protección

Un sistema computacional es una colección de procesos y objetos.

Objetos: CPU, segmentos de memoria, discos, etc. Programas. Cada objeto es distingible de los demás objetos. Un objeto es un tipo abstracto de datos.

Las operaciones posibles dependen de los objetos. Un objeto CPU sólo puede ejecutar procesos. Un proceso tiene permitido accesar sólo aquellos recursos que están autorizados para su uso.

Ejemplo: un compilador no tiene acceso a cualquier archivo: sólo aquellos que están bien definidos y pueden ser compilados. El compilador usa sus propios archivos que no pueden ser accesados por otros procesos.

5.3.1 Estructura de dominio

Un proceso opera dentro de un dominio de protección que especifica los recursos que puede acceder. Cada dominio define un conjunto de objetos y tipos de operaciones que pueden ser invocadas por cada objeto. La posibilidad de ejecutar una operación sobre un objeto se denomina **derecho de acceso**.

Un dominio es una colección de derechos de accesos, cada uno de los cuales es el par: **<nombre de objeto, conjunto de derechos>**.

El dominio D tiene derechos de acceso:

ej: <archivo F, {lectura,escritura}>

Un proceso ejecutándose en el dominio D puede leer y escribir el archivo F, pero no puede hacer ninguna otra operación sobre este objeto.

Los dominios no necesitan ser disjuntos. Pueden compartir derechos de acceso.

Gráfica 86. Dominios compartidos

El derecho de acceso <O4, {print}> es compartido por D2 y D3. Esto implica que un proceso que se ejecute en cualquiera de estos dominios puede imprimir el objeto O4

La asociación entre un proceso y un dominio puede ser *estática* o *dinámica*.

La implementación más fácil es la estática.

Cuando es dinámica, es necesario un mecanismo para permitir que un proceso conmute de un dominio a otro.

5.3.2 Niveles de dominios

Un dominio puede ser:

- **Cada usuario.** El conjunto de objetos que se pueden accesar dependen de la identidad del usuario. La conmutación ocurre cuando se cambia el usuario.
- **Cada proceso.** Los objetos a acceder dependerán de la identidad del proceso. La conmutación de dominios ocurre cuando un proceso envía un mensaje a otro, esperando por respuesta.
- **Cada procedimiento.** El conjunto de objetos corresponde a las variables locales definidas en el procedimiento. La conmutación de dominios ocurre en una llamada a procedimientos.

5.3.3 Modo dual de operación

Un ejemplo de dominio es el modo dual de operación de la CPU:

- Modo usuario.
- Modo kernel o monitor.

Un proceso que se ejecuta en modo usuario sólo tiene acceso a instrucciones no privilegiadas. En un sistema multiprogramado esto es insuficiente.

5.4 Matriz de acceso

El modelo de protección puede ser visto en forma abstracta como una **matriz de acceso**.

Las filas representan dominios y las columnas objetos.

La matriz de acceso permite implementar una variedad de políticas. El mecanismo consiste en implementar la matriz de acceso y asegurar que un proceso que se ejecuta en el dominio **D_i** pueda accesar sólo los objetos especificados en la fila *i*, en la forma como se especifica en las entradas de la matriz.

Gráfica 87. Matriz de acceso

Objeto	F1	F2	F3	Printer
Dominio				
D1	read		read	
D2				print
D3		read	ejecutar	
D4	read write		read write	

5.4.1 Implementación de políticas

La matriz de acceso provee un mecanismo que permite especificar una variedad de políticas.

Decisiones políticas relativas a protección se pueden implementar con la matriz de acceso, indicando qué derechos se deben incluir en la entrada (i,j). Además se debe decidir el dominio en el cual cada proceso se ejecuta (usualmente esta política la decide el S.O.).

Los usuarios normalmente deciden el contenido de las entradas de la matriz de acceso. Cuando se crea un nuevo objeto **O_j**, la columna **O_j** se agrega y la entrada se fija acorde a los derechos que dicta el creador.

Hay que notar que la matriz de acceso también es un objeto. Cuando se cambia su contenido, se están haciendo operaciones sobre ella. Se debe proteger cada entrada de ella.

5.4.2 Conmutación de dominios

La conmutación es también un derecho. Si **Di** conmuta a **Dj**, la entrada **(i,j)** de la matriz de acceso, debe contener el derecho **switch**.

Por ejemplo, un proceso ejecutándose en el Dominio **D2** puede conmutar al Dominio **D3** o el Dominio **D4**.

Gráfica 88. Matriz de acceso con dominios como objetos

Objeto	F1	F2	F3	Print er	D1	D2	D3	D4
Dominio								
D1	read		read			S		
D2				print			S	S
D3		read	ejecu tar					
D4	read write		read write		S			

La entrada S significa switch, o sea, la posibilidad de conmutar.

5.5 Seguridad

La protección es un problema interno. La seguridad incluye además el ambiente externo.

Ejemplo. La "seguridad ciudadana" es más que tener alarma, rejas o guardias en los barrios.

La seguridad en S.O es esencialmente un problema de administración.

¿Qué se saca tener buenos sistemas de protección si cualquiera puede llevarse un disco removible?

5.5.1 El problema de la seguridad

Un sistema es seguro si sus recursos se usan y acceden acorde a la forma como están definidos en toda circunstancia. Esto no es posible siempre.

Algunas violaciones de seguridad pueden ser:

- Intencionales

- Lectura no autorizada de datos
- Modificación no autorizada de datos
- Destrucción no autorizada de datos
- Accidentales

La protección absoluta no es posible. Se busca subir el costo de violaciones de seguridad poniendo barreras adecuadas.

5.5.2 Medidas de seguridad

Para proteger un sistema, se deben tomar medidas en dos niveles diferentes:

- **Físico.** Los lugares que contienen el sistema deben tener protección de entradas a personas ajenaas.
- **Humano.** Asegurar que un usuario no de acceso a una persona no autorizada. Esto es difícil.

Es importante para cualquier organización proteger sus datos: información para la competencia ("*La casa del espía*")

5.6 Autenticación

El principal problema de seguridad para un S.O es la **autenticación**. Es necesario proteger los programas y los procesos que se están ejecutando.

¿Cómo saber si un usuario que se identifica es auténtico?. Para hacer esto se usan tres cosas:

- **Algún objeto:** llave o tarjeta.
- **Algún conocimiento:** Password.
- **Algún atributo:** huellas digitales, imagen de la retina, largo de los dedos.

5.6.1 Password

Las password se usan para proteger objetos: archivos, cuentas, etc. Distintos derechos de acceso se protegen con passwords diferentes.

Dificultad: ¿cómo mantenerla secreta?

Encriptación de password

UNIX usa un sistema de encriptación para mantener las passwords secretas. Cada usuario tiene una password, por ejemplo **x**.

El sistema tiene una función $f(x)$. El sistema almacena $f(x)$. Obtener x a partir de $f(x)$ en la práctica resulta imposible. Este método hace posible que el archivo de password no sea secreto.

5.7 Amenazas

Existen dos tipos de amenazas:

- A programas y
- Al sistema.

En ambientes donde un programa escrito por un usuario puede ser usado por otro, se presenta un problema de seguridad.

5.7.1 Amenazas a programas

Pueden ocurrir dos situaciones, en cuanto a amenazas a programas:

- Caballo de Troya
- Trampas

Amenazas a programas: Caballo de troya

¿Que pasa si un programa escrito por un usuario es usado por otro?. El programa podría usar de mala manera los derechos del usuario que ejecuta.

Ejemplo. Editor de texto escrito por Verónica es usado por Carolina. El editor busca palabras claves y si las encuentra copia el texto completo en un archivo de Verónica.

Un segmento de código que usa mal un ambiente se llama caballo de troya. Una variación es un programa que emula un ambiente de login. W/NT evita esta situación con secuencia Ctrl-Alt-Del.

Amenazas a programas: Trampas

Una programadora puede dejar un "hoyo" en un programa que sólo ella es capaz de usar.

Ejemplos reales son programas bancarios que tienen errores de redondeo. Los centavos son todos transferidos a alguna cuenta.

Encontrar trampas es muy difícil por la gran cantidad de líneas de código que puede llegar a tener una aplicación.

5.7.2 Amenazas a sistemas

La mayoría de los sistemas operativos proveen de métodos para que procesos puedan generar otros procesos (ej. fork() en UNIX).

En estos sistemas, es posible crear situaciones donde los recursos y archivos de usuarios sean mal usados.

Los métodos más comunes son:

- Gusanos (Worms)
- Virus

Amenazas al sistema: Gusanos

Un **gusano** es un proceso que usa el mecanismo de generación de procesos para afectar el rendimiento de un sistema.

Un gusano genera copia de sí mismo sobre una red de computadores.

En 1988 ocurrió este evento sobre Internet causando pérdidas de millones de dólares. Robert Tappan, un aventajado estudiante de sistema operativos de la U. de Cornell fue el autor de este maléfico gusano.

Amenazas al sistema: Virus

Los virus son diseñados para infectar programas y producir daños en el sistema, modificando y destruyendo archivos, generando caídas y mal funcionamiento.

Un gusano es un programa completo. Un virus es un fragmento de código oculto en un programa inocuo.

Normalmente el ataque se produce entre usuarios de PC. Los sistemas multiusuarios son menos vulnerables ya que los programas ejecutables se protegen contra escritura. Aún si se infectan programas otros componentes están bien protegidos.

Infección por Virus

La forma de contagio es vía bajar programas infectados de la red o copiar desde disquettes.

Para evitarlos se deben manejar criterios como el de computación segura y cuidar sector de booteo. Controlar el checksum de cada archivo.

Los Virus son más bien problemas de seguridad que de protección.

5.7.3 Monitoreo de amenazas

El monitoreo es una técnica de administración.

¿Qué monitorear?

- En un sistema tiempo compartido: nº de password incorrectas.
- Auditar bitácoras de sistema: usuarios conectados, tiempo de conexión.

Indudablemente los sistemas de red son mucho más susceptibles a ataques de seguridad. El tema de seguridad en redes se debe considerar dentro de la seguridad en sistemas operativos.

5.8 Encriptación

Cualquier sistema mantiene datos sensibles para la organización. A través de las redes viaja esta información. Es posible entonces accesar o interceptar información sensible.

Para proteger información sensible se necesitan mecanismos de protección.

5.8.1 Mecanismos de encriptación

El mecanismo básico trabaja de la siguiente manera:

- La información (*texto*) es encriptada desde su forma legible (*texto claro*) a una forma interna (*texto cifrado*). El texto cifrado no es posible de entender.
- El texto cifrado se puede almacenar en archivos sin protección de lectura o transmitido sobre canales inseguros.
- Para leer el texto cifrado se necesita desencriptar la información para volverla a transformar en texto claro.

5.8.2 Propiedades de la encriptación

Sea:

- **E**: algoritmo general de encriptación
- **D**: algoritmo general de desencriptación
- **k**: clave de encriptación
- **m**: mensaje

Se debe cumplir que:

- $D_k(E_k(m)) = m$
- E_k y D_k se calculen en forma eficiente
- Si se publica E, no hay forma de calcular D
- La seguridad debe depender de k y no de los algoritmos E y D

5.8.3 Algoritmos de encriptación

Una forma posible para realizar una **encriptación** es el estándar DES (**Data Encryption Standard**)

¿Cómo distribuir la clave k?

- Una solución posible es la utilización de claves públicas y claves privadas.
- Cada usuario tiene dos claves una pública y una privada. Ambos se pueden comunicar conociendo sólo la clave pública del otro.

Algoritmo RSA (Rivest, Shamir y Adleman): ejemplo de algoritmo de claves públicas

- Clave pública: $\langle E, n \rangle$
- Clave privada: $\langle D, n \rangle$
- E, D, n : enteros positivos
- Cada mensaje se representa como un entero entre 0 y $n-1$ (mensajes largos se descomponen en mensajes cortos. Cada uno se representa por un entero).
- Se definen E y D como:
- $E(m) = m^E \text{ mod } n = C$
- $D(C) = C^D \text{ mod } n = m$

Propiedades del algoritmo RSA

Sea:

- E:** Llave pública para codificar
- D:** Llave privada para decodificar
- m:** mensaje

Se debe cumplir que:

- 1) $D(E(m)) = m$
- 2) E y D se calculan en forma fácil
- 3) Si se publica E , no hay forma de calcular D
- 4) $E(D(m))=m$

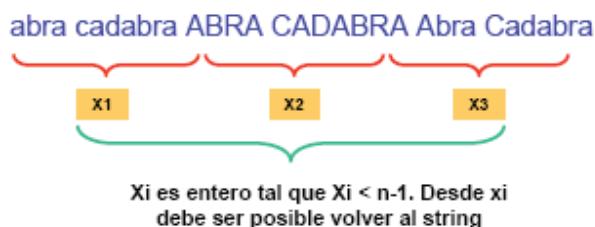
¿Cómo se obtiene D y E ?

- Hay que obtener p y q : números enteros, primos y muy grandes.
- Se calcula $n = p \times q$.
- Se elige $E < n$ tal que E y $(p-1)(q-1)$ son primos relativos.
- Se calcula D tal que $E \times D \text{ mod } ((p-1)(q-1)) = 1$
- La Llave pública es $\langle E, n \rangle$
- La Llave privada es $\langle D, n \rangle$

Gráfica 89. Codificación RSA

Codificación RSA

- Dado un texto, se separa en grupo de caracteres de tamaño fijo y se transforman a números enteros.
- Por ejemplo:



Codificación RSA

- **Codificar:** $x_i^E \bmod n = y_i$
- **Decodificar:** $y_i^D \bmod n = x_i$
- **Ejemplo:**

Sea $p=2$ y $q=5$ (p y q son primos)
Sea $n=10$
$(p-1)x(q-1)=4$
$E < n$ y 4 deben ser primos relativos. Elijo $E=3$
Cálculo de D : $E \times D \bmod ((p-1)(q-1))=1$
O sea, $E \times D \bmod 4 = 1$, $E \times D = 4k + 1$
$E \times D = 4k + 1$. ¿Existe k ?
$3 \times D = 4k + 1$. $D = (4k+1)/3$. Si $k=2$, $D=3$. Si $k=5$, $D=7$
21 mod 10=1 ¡Funciona!
Sea $x=2$ el número a encriptar. $2^E=2^3=8 \bmod 10=8$, $y=8$.
Ahora desencriptaremos. $8^D=8^3=2097152 \bmod 10=2$ ¡Funciona!

¿Cómo se eligen las claves?

La seguridad depende del número de bits:

- Netscape y Explorer usan RSA con 48 bits.
- Mejor seguridad se obtiene con 128 y 1024 bits, pero el cálculo se vuelve muy difícil.

¿Se puede quebrar RSA?

En 1977 se lanzó el reto de quebrar un string de 129 dígitos (430 bits). Se pensaba que era inviolable ya que los algoritmos de factorización de grandes números estimaban 40 cuadrillones de años de computación para quebrarlo.

En abril de 1994, sólo 17 años más tarde, cuatro científicos reportaron que habían quebrado el código. El mensaje encriptado era:

THE MAGIC WORDS ARE SQUEAMISH OSSIFRAGE

Se utilizó un método de factorización que requería 5000 MIPS-año. Se dividió el problema en pequeñas partes que se distribuyeron por correo electrónico a todo el mundo. En la práctica no se requieren 5000 MIPS-año ya que las claves no siempre se eligen de manera óptima.

5.9 Clasificación de seguridad

El Departamento de Defensa del Gobierno de USA, especifica 4 divisiones para clasificar sistemas acorde a su seguridad:

A (protección máxima), B, C, D (protección mínima).

Por ejemplo MS-DOS corresponde a la clasificación D.

El nivel C tiene dos niveles:

- **C1:** protege información privada y resguarda a los demás usuarios de lectura o destrucción accidental de información. Ej. Muchas implementaciones de Unix.
- **C2:** Agrega a C1 de un control de acceso a nivel individual. Existen versiones seguras de Unix certificadas como C2. W/NT se puede configurar a C2.

El nivel B tiene todo lo de C2 y agrega rótulos de sensibilidad a cada objeto. Se clasifican en B1, B2 y B3.

El nivel más alto es A. Al nivel B3 agrega especificaciones de diseño y verificación formal de seguridad.

Actividad inicial:

De acuerdo a lo desarrollado en el capítulo, revise su ensayo y compleméntelo, sería bueno que coevaluará su trabajo con un compañero de clase.

Los aportes enriquecen el producto obtenido..

BIBLIOGRAFÍA

ALCALDE, Eduardo. MORERA, Juan. PEREZ-CAMPANERO, Juan A. (1994). *Introducción a los Sistemas Operativos. Serie Informática de Gestión.* México: Editorial Mc Graw Hill.

BARRETO ROA, Julio Humberto. (2001). *Sistemas Operativos. Guía de estudio.* Bogotá: Editorial UNAD.

CALDERA (2003). *Kit de recursos. Unifying Unix Whit Linux For Business.*

CAÑAS, Javier. Documento pdf: Sistemas Operativos. Catorce capítulos (1999).

CARRETERO PEREZ, Jesús, GARCIA CABALLEIRA, Félix, ANASAGASTI, Pedro de Miguel, PEREZ COSTOYA, Fernando (2001). *Sistemas Operativos. Una visión aplicada.* Madrid: Mc Graw Hill.

FLYNN, Ida M, MCCHOES, Ann McIver.(2001) *Sistemas operativos. Tercera Edición.* México: Editorial Thomson Learning.

RAYA, Laura, ALVAREZ, Raquel, RODRIGO, Víctor. (2005). *Sistema Operativos en entornos Monousuario y Multiusuario.* México: Alfaomega, Ra-Ma.

RUEDA, Francisco. (1989). *Sistemas Operativos.* Santafé de Bogotá: Mc Graw Hill.

SILBERSCHATZ, Avi, GALVIN, Peter, GAGNE, Greg. (2002). *Sistemas Operativos.* México: Editorial Limusa Wiley.

STALLING, William. (2001). *Sistemas operativos. Cuarta edición,* México: Prentice Hall.

TACKETT, J. (2003). *Edición especial Linux.* México: Prentice Hall

TANENBAUM, S. Andrew, WOODHULL, Albert S. (1997). *Sistemas Operativos. Diseño e implementación.* México: Prentice Hall.

DIRECCIONES WEB

<http://www.tau.org.ar/base/lara.pue.udlap.mx/sistoper/>

<http://www.itver.edu.mx/so1/>

<http://www.itver.edu.mx/so2/>

<http://os.matiu.com.ar/>

<http://os-matiu.dreamhost.com/classes/clase1.html>

<http://www.iespana.es/canalhanoi/so/>

http://server2.southlink.com.ar/vap/sistemas_operativos.htm

<http://www.inei.gob.pe/web/metodologias/attach/lib616/INDEX.HTM>

<http://www.itq.edu.mx/vidatec/maestros/sis/mnogues/Unidad1.htm>

<http://www.cs.virginia.edu/~knabe/iic2332/notes01.html>

UNIDAD DIDÁCTICA 3

PRINCIPALES SISTEMAS OPERATIVOS

INTRODUCCIÓN

Sistemas operativos hay muchos, según la necesidad, la potencia del hardware, la seguridad que se desea, el respaldo y confiabilidad. Cada uno de ellos ha evolucionado en diferentes versiones que hacen que la actual siempre supere por mucho a la anterior y así ganarse un merecido puesto en el tan competitivo mercado del software.

En esta unidad vamos a enfocarnos en los dos sistemas operativos actuales más comerciales y conocidos: Windows y Linux, sin dejar de lado otros no menos importantes, además de algunas características de las principales arquitecturas para trabajo en red.

Aquí se vuelven muy importante los criterios que se deben tener en cuenta a la hora de seleccionar el “**sistema operativo ideal**”, pues de ello depende la conformidad de nuestro usuario final, que en últimas es el que “**sufre**” las consecuencias de nuestra decisión.

Pero ya que menciono el “sistema operativo ideal”, es bueno aclarar que no existe, pues aquello que llamo “**ideal**” se logra de acuerdo al grado de satisfacción de nuestros clientes y el grado de optimización logrado en el sistema que está o que se va a montar.

Ahí es donde entran en juego todas las características vistas en las dos unidades anteriores, que se deben analizar muy bien en cada uno de los sistemas operativos que se están “analizando” para su implementación.

Cada uno tiene puntos a favor y puntos en contra, en mi concepto, y sólo en mi concepto, digo que no existe un sistema operativo que se pueda recomendar a la ligera, sin antes haber determinado las necesidades, tipos de aplicaciones, nivel de seguridad, costos, etc, para su implementación.

Sin embargo tengo mi favorito, que no estaría bien mencionarlo.

Para ubicarnos en la diversidad de sistemas operativos que existen, a continuación se presenta la siguiente tabla que contiene sistemas operativos, según la casa matriz que lo desarrolla:²⁹

AtheOS/Sybble/Cosmoe	Familia UNIX
OSBOS	AIX
PenBeOS	AMIX
Zeta	GNU/Linux
BlueeyedOS	GNU / Hurd
Cosmoe	HPUX
BeFree	Irix
Sequel	Minix
SkyOS	System V
Familia Amiga	Solaris
AmigaDOS 1.x	UnixWare
AmigaOS 2	LynxOS
AmigaOS 3.03.1	Xenix
AmigaOS 3.5/3.9	Familia BSD
WarpOS (AmigaOS 3.x + subsistema PowerPC)	FreeBSD
MorphOS	NetBSD
AmigaOS 4.0	VINO
Familia Macintosh	OpenBSD
Mac OS 8	PicoBSD
Mac OS X	Familia Mach
Familia QNX	GNU / Hurd
RTOS	BSD lites
Neutrino	Mac OS X
RTP	NEXTSTEP
Familia DOS	YAMIT
MSDOS	MKlinux
DRDOS	Familia IBM
PCDOS	OS/2
FreeDOS	OS/360
Novell DOS	OS/390
Familia Microsoft	OS/400
Microsoft XP	Sist. Op. Académicos o experimentales
Windows 95, 98, Me	Chorus/Jaluna
Windows NT	Amoeba
Windows CE	MIT Exokernel
Windows 2003	BriX
Windows Longhorn?	

²⁹ GUARQUIN, Margarita y QUIROGA, Edgar (2005). Módulo ensamblaje y mantenimiento de computadores. Bogotá: UNAD.

OBJETIVOS

1. Reconocer las diferentes características internas y externas de los sistemas operativos de la familia Windows, con el fin de obtener el máximo provecho posible en su administración.
2. Identificar la estructura interna de los sistemas operativos de la familia UNIX (UNIX/LINUX), para entender su funcionamiento y establecer las ventajas y desventajas con respecto a otros sistemas del mercado.
3. Explorar otros sistemas operativos, que aunque es complicado trabajar con ellos en el laboratorio, es necesario conocer su estructura interna y características más sobresalientes.
4. Revisar los conceptos básicos de las diferentes arquitecturas de red, en las que puede operar un sistema operativo.

CAPÍTULO 1. SISTEMAS OPERATIVOS FAMILIA WINDOWS³⁰

Actividad a realizar: En grupos de trabajo. En sesiones de laboratorio o prácticas se debe instalar el respectivo sistema operativo y verificar las ventajas y características básicas que este posee. A medida que se va desarrollando la temática Usted debe ir detectando cuáles características de cada SO Windows ya conocía y cuáles son nuevas, para poder verificarlas.

Realizar un manual de instalación de la versión del SO. Probarlo en la sesión.

Como se ha descrito en la primera unidad, existen distintos tipos de sistemas operativos. Hay dos tipos que son especialmente necesarios (si no imprescindibles) para los usuarios:

- **Multitarea:** Es el modo de funcionamiento mediante el cual un computador procesa varias tareas al mismo tiempo.
- **Multiusuario:** Se pueden dar dos opciones:
 - **De servidor de red:** que permite cumplir simultáneamente las necesidades de dos o más usuarios, para trabajar con los mismos o distintos recursos del equipo.
 - **De estación de trabajo:** que permite que cada usuario pueda tener sus propias carpetas, archivos, escritorio, historial, directivas de seguridad, etc., sin interferencias con los de otros usuarios. Esta opción permitirá que cada usuario que trabaje en un ordenador, tenga un mínimo de seguridad para que sus datos estén a salvo de las miradas indiscreta del resto de usuarios que trabajen en el mismo equipo.

Este capítulo se va a centrar en una exploración de los principales sistemas operativos de la familia Windows, iniciando con el primero de la línea de entorno gráfico Windows 95 hasta llegar a Windows Server 2003. De esta forma se pretende brindar una visión general de la evolución que ha sufrido hasta el momento la línea de sistemas operativos Windows, tanto en su estructura como en su diseño.

Se va a hacer un énfasis especial en los dos sistemas operativos multitarea y multiusuario: **Windows Server 2003** que funciona como servidor de red cuando se instala el directorio activo y **Windows XP Profesional** que funciona como estación de trabajo.

³⁰ La documentación referida en este capítulo sobre la Familia Windows, fue tomada de varias fuentes básicas como son:

1.1 Windows 95

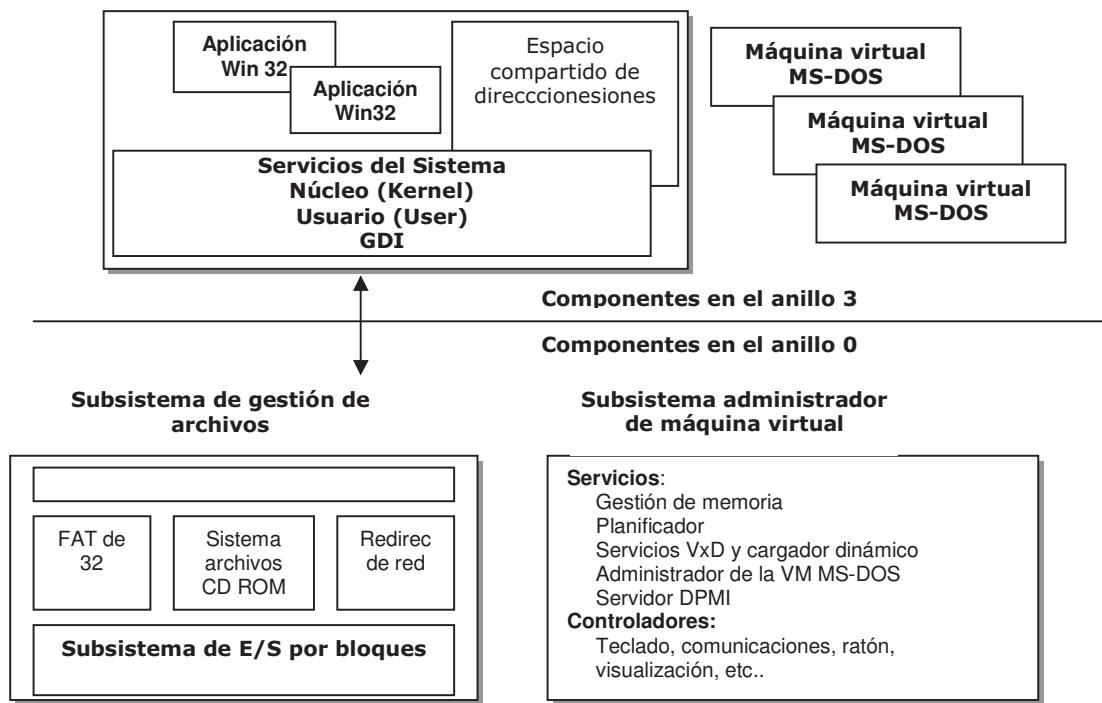
1.1.1 Características

- Sistema operativo orientado a documentos.
- El sistema tiene asociado un programa con cada fichero de datos, arranca de manera automática la aplicación asociada. (Por ejemplo abrir un texto de WORD picando en fichero, sin abrir previamente la aplicación).
- Sistema operativo de 32 bits (con lo que se pueden aprovechar mejor las características de los nuevos procesadores); si se trabaja con aplicaciones de 16 bits no se le saca el rendimiento a la multitarea.
- Windows 95 ofrece un soporte muy completo para las prestaciones multimedia que se puede añadir a cualquier PC de casa.
- Multitarea: el sistema es capaz de ejecutar más de un programa al mismo tiempo.
- Aprovecha los tiempos muertos para ejecutar instrucciones de otros procesos (por ejemplo mientras el controlador del disco lee un bloque de datos).
- Permite abrir varias sesiones de DOS con la ventana del MS-DOS.
- La interfaz de usuario es más fácil de usar.
- Se pueden usar nombres largos de fichero, con lo que no se limita a 8 caracteres para el nombre y 3 para la extensión. Ahora pueden tener un tamaño máximo de 256 caracteres.
- Elemento nuevo son los accesos directos que son enlaces a un fichero que está en un directorio determinado, de forma que se puede acceder a dicho fichero picando dos veces en el acceso directo situado en el escritorio en vez de tener que colocarnos en el directorio en el que se encuentra dicho fichero.
- La pantalla general de Windows 95 permite acceder a todos los elementos tanto de software como de hardware del sistema. Partes de la pantalla:
 - Escritorio
 - Barra de tareas
- Todos los programas se ejecutan en una o varias ventanas, salvo los que usan la pantalla completa (juegos).
- El rendimiento es FAT 16, de Windows 95.
- El disco duro con el formato FAT 16, no se podrá un disco con otro formato a no ser que se vuelva a realizar un nuevo formateo de la unidad bajo ese formato.
- Tiene varios inconvenientes, como configurar infinidad de parámetros, localizar los drivers correctos e instalar las nuevas versiones de los programas. La mayoría de las aplicaciones funcionaban más lentamente que en MS-DOS y Windows 3.1, dado que no aprovechaban las mejoras realizadas.

1.1.2 Arquitectura

Gráfica 90. Arquitectura Windows 95

Fuente: www.club.telepolis.com



Máquina Virtual del sistema

- Servicios de planificación y gestión de memoria.
- Gestión de las aplicaciones basadas en Windows dentro de la máquina virtual del sistema.
- Gestión de las máquinas virtuales MS-DOS
- Las bases de la capa API de Windows. (recordemos que la capa API, es la interfaz que pueden utilizar las aplicaciones para comunicarse con Windows).

Todas las aplicaciones Windows, se ejecutan dentro del contexto de la VM (*Virtual Machine*) del sistema. Las aplicaciones de 16 bits (las “viejas” aplicaciones de Windows 3.1) comparten un espacio único de direcciones. El soporte de 32 bits, proporciona a cada aplicación un espacio de direcciones privado.

Máquinas virtuales MS-DOS. Windows 95, admite la ejecución de múltiples programas MS-DOS que se ejecutan bien en modo virtual 8086 o bien en modo protegido.

Administrador de máquina virtual. El VMM (Administrador de Máquina Virtual) es el verdadero corazón del sistema operativo. Proporciona servicios de planificación y gestión de memoria de bajo nivel, además de los controladores virtuales de dispositivo.

De todos los componentes nuevos que aparecen en Windows 95, el principal es el **sistema de gestión de archivos**. Este es un subsistema completamente rediseñado que admite múltiples sistemas de archivos accesibles concurrentemente. Excepto cualquier viejo controlador de dispositivo MS-DOS, que pudiera estar presente para soportar un dispositivo particular, todo el Sistema de gestión de archivos ejecuta código de 32 bits en modo protegido. Su diseño original, admitía discos locales y unidades CD-ROM, además de una o más interfaces de red mediante una interfaz del sistema de archivos instalable (IFS).

Los servicios del sistema llamados por las aplicaciones Windows (para gráficos, gestión de ventanas, etc) siguen estando allí y conservan los nombres de Núcleo, Usuario y GDI que tenían en las versiones anteriores de Windows. El cambio principal en el subsistema de servicios del sistema es su soporte para las aplicaciones de 32 bits. Aparte de sus diferentes requisitos de gestión de memoria las aplicaciones de 32 bits utilizan una API completa de 32 bits de Windows y llaman a servicios que ahora se implementan con la utilización del código de 32 bits. Uno de los principales desafíos de diseño e implementación con los que se enfrentó el equipo de desarrollo de Windows fue hacer que cooperasen de forma efectiva y con un buen rendimiento (al menos aceptable) los componentes de 16 y 32 bits mezclados.

Manejo de aplicaciones

Las aplicaciones que se pueden instalar en Windows'95 se pueden clasificar en dos grupos:

- **Los programas autónomos**, que sólo hay que copiarlos en el disco duro para que funcionen.
- **Las aplicaciones con instalación**, que necesitan un proceso previo para funcionar que consta de una serie de programas y bibliotecas dinámicas (.dll). Estas aplicaciones contienen un programa (instalar, setup) que realiza estas funciones.

Manejo de comunicaciones

Hyper Terminal

Es un sistema básico de comunicación para conectarse a otros sistemas. Es un emulador de terminal tipo texto y sólo se puede utilizar para conectarnos con un sistema multipuesto y multitarea (ej. UNIX). También se pueden transferir ficheros de una máquina a otra si se dispone del software apropiado.

Acceso telefónico a redes

Su función principal es permitir el acceso a Internet. El protocolo de comunicación entre los equipos que componen Internet se llama TCP/IP, sin embargo al no estar nuestro equipo físicamente conectado a la red hay otro protocolo que nos permite conectarnos, que se llama PPP.

1.2 Windows 98

1.2.1 Características

- Un núcleo de sistema de 32 bits que incluye gestión de memoria, multitarea preentiva y soporte de multitarea.
- Un sistema de archivos en modo protegido de 32 bits, que elimina la necesidad de contar con una copia separada de MsDos una vez que el sistema arranca.
- Un sistema de ficheros de 32 bit que soporta FAT, FAT32, ISO 9660 (CDROM), ISO 13346 (UDF/DVD – Universal Disk Format / Digital Video Disk), redireccionamiento de redes y alta eficacia. Este sistema de ficheros también soporta el uso de nombres largos de fichero.
- Soporte para WDM (Win32 Driver Model), que hace que los dispositivos que cuentan con este driver puedan ejecutarse, usando el mismo driver, en Windows 98 y Windows NT.
- Chequeo automático del sistema cada vez que una aplicación falla.
- Un entorno de configuración dinámico que reduce la necesidad de ajustes y reinicios del sistema manualmente por parte del usuario.

1.2.2 Arquitectura

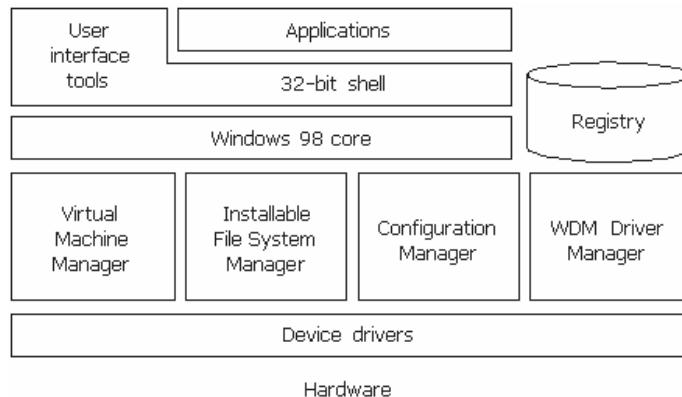
Windows 98 es un sistema operativo de 32 bits que incorpora acceso interno a Internet, soporte de hardware Plug & Play y compatibilidad con Windows 95.

Como mejoras frente a la anterior versión de Windows 95 incluye mejoras en la gestión de energía, soporte para múltiples tarjetas de vídeo, y soporte para hardware aparecido recientemente.

Los principales componentes de Windows 98 son los siguientes:

- Aplicaciones.
- Interfaz de usuario e interfaz de órdenes de 32 bits. Base de Registro.
- Núcleo de Windows.
- Administrador de máquina virtual. Administrador del sistema de archivos instalable (IFS). Administrador de configuración. Administrador WDM.
- Controladores de dispositivo.

Gráfica 91. Componentes de Windows 98



Drivers de dispositivos (Device Drivers)

Windows 98 proporciona soporte mejorado para dispositivos de hardware, incluyendo tarjetas gráficas, unidades de disco, módems, fax, impresoras, etc.

En Windows 3.1 los drivers (controladores para el dispositivo, normalmente creados por el fabricante del propio dispositivo) eran, en la mayoría de los casos, monolíticos y difíciles de desarrollar, ya que el driver tenía que proporcionar todos los servicios, interfaces de usuario, funciones API y servicios de acceso al hardware.

Con Windows 98 se pasa a una arquitectura basada en **minidrivers**, que hace que el desarrollo del driver sea más fácil para los fabricantes: el sistema operativo proporciona los servicios básicos para las distintas clases de dispositivos de hardware (lo que se denomina un **driver genérico o universal**) y el fabricante sólo tiene que aportar los códigos específicos para su hardware particular (minidriver). Además Windows 98 utiliza arquitectura WDM, que proporciona un juego común de servicios de entrada/salida compatible también con Windows NT, de modo que los drivers que se desarrolle servirán en ambos sistemas operativos.

El driver virtual (VxD)

Un driver virtual (VxD) es un driver de 32 bits que funciona en modo protegido y que administra recursos del sistema tales como dispositivos de hardware o software instalado, de modo que más de una aplicación pueda usar el recurso al mismo tiempo. Es ejecutado en modo kernel y tiene todos los niveles de privilegio para efectuar cualquier operación.

Windows 98 carga dinámicamente estos VxDs en memoria siempre que son necesarios. Igualmente, el sistema operativo utiliza para ellos memoria "bloqueada". Es decir marcada como no paginable, para que permanezcan siempre en memoria e intentar optimizar así los accesos a estas VxD.

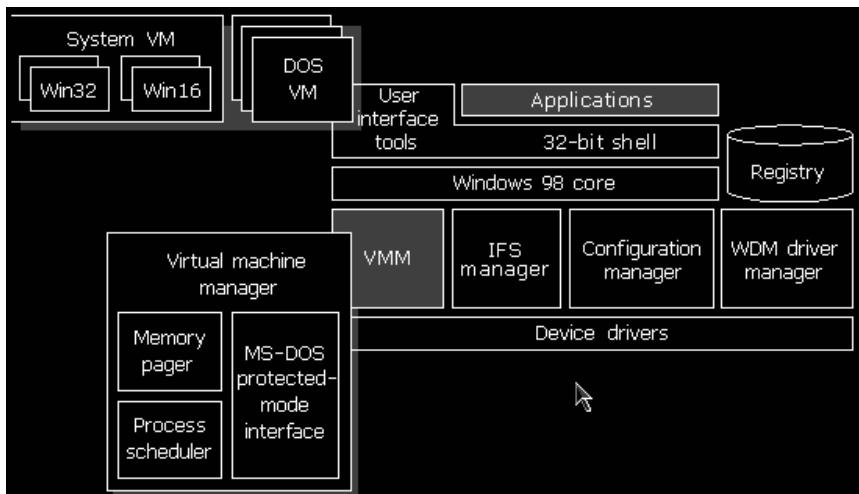
Un driver VxD realiza un control sobre el estado del recurso que está bajo su gestión, para cualquier aplicación que desee usar dicho recurso. La misión del VxD es comprobar el estado del recurso y actuar como árbitro entre las aplicaciones que intentan acceder a él. Además asegura que el recurso funciona adecuadamente, y toma el control cuando algo falla.

Gestor de Máquina Virtual (VMM)

El gestor de máquina virtual, gestiona todos los recursos necesarios para cada aplicación y proceso que se ejecute en el ordenador. El gestor de máquina virtual crea y mantiene un entorno para cada aplicación, en el que dicha aplicación y el sistema se ejecutan correctamente.

Una **máquina virtual** (VM), es un entorno creado en memoria que, desde el punto de la aplicación, aparece como un ordenador separado, con todos los recursos disponibles del ordenador físico que la aplicación necesite para ejecutarse.

Gráfica 92. Recursos que necesita el sistema para VM



Windows 98 tiene un único sistema denominado sistema VM, en el que todos los procesos del sistema se ejecutan.

El gestor de máquina virtual es responsable de tres áreas de servicios:

- Planificador de procesos (para permitir la multitarea).
- Página de memoria.

- Soporte protegido para modo MsDOS (solo para aplicaciones basadas en MsDOS).

El planificador de procesos de Windows 98 utiliza dos métodos para permitir que varios procesos se ejecuten de forma concurrente: multitarea cooperativa y multitarea con derecho preferente o preventiva. (Recordemos Unidad didáctica 2. Capítulo 1. Tema 1.5.4 Planificación apropiativa o no apropiativa)

La multitarea cooperativa. Esta depende de la ayuda de los programadores de aplicaciones para mantener el sistema ejecutándose sin sobresaltos.

Con la técnica cooperativa el planificador puede commutar entre procesos sólo cuando el proceso actualmente en ejecución entrega la CPU. Según la buena práctica de programación para sistemas de multitarea cooperativa las aplicaciones deben devolver regularmente la CPU al sistema operativo.

Es la que se utiliza con las aplicaciones de Windows 3.1 (aplicaciones Win16), y se mantiene en Windows 98 para mantener la compatibilidad con dichas aplicaciones.

Multitarea preventiva. Esta planificación pone a disposición del Windows 98 un control completo sobre qué proceso se ejecuta a continuación y por cuánto tiempo. En cualquier momento, el planificador puede despojar de la CPU al proceso en curso y asignársela a otro.

Sistema de archivos instalable (IFS)

Windows 98 presenta una arquitectura en capas del sistema de ficheros que soporta múltiples sistemas de ficheros, incluyendo los VFAT (virtual file allocation table), CDFS (CDROM file system), UDF, etc.

Esta arquitectura nos permite usar nombres largos de archivo y un sistema de caché dinámico para entrada/salida en ficheros y redes. Se pueden emplear nombres de hasta 255 caracteres para identificar los documentos en lugar de la relación 8.3 de MsDOS. Además los nombres de los ficheros son menos encrípticos y fáciles de leer, porque Windows 98 esconde los nombres de las extensiones a los usuarios.

La arquitectura de ficheros de Windows 98 se basa en tres componentes:

- Installable File System (IFS) Manager. Es el responsable de arbitrar los accesos a los distintos componentes de los sistemas de ficheros.
- File System Drivers. Incluye acceso a discos con particiones FAT, ficheros en CDROM, y redirección de dispositivos de red.

- Block I/O Subsystem. Es responsable de interactuar con los dispositivos de hardware.

Sistema VFAT (FAT 32)

Este sistema de ficheros es la novedad que Windows 98 presentó en su lanzamiento. Es un sistema de ficheros que presenta una serie de ventajas sobre los antiguos sistemas FAT, usados por DOS.

- Mayor velocidad de acceso dado que el software que se utiliza para la caché de disco trabaja en modo real de 32 bits.
- No utiliza memoria convencional.
- Permite mejorar los accesos en multitarea al disco.
- Soporte de caché dinámico.
- Utilización optimizada para accesos en 32 bits.

Windows 98 incluye la utilidad gráfica de conversión Convertidor a FAT32, que convierte de modo rápido y seguro discos duros de FAT a FAT32.

Gestor de configuración

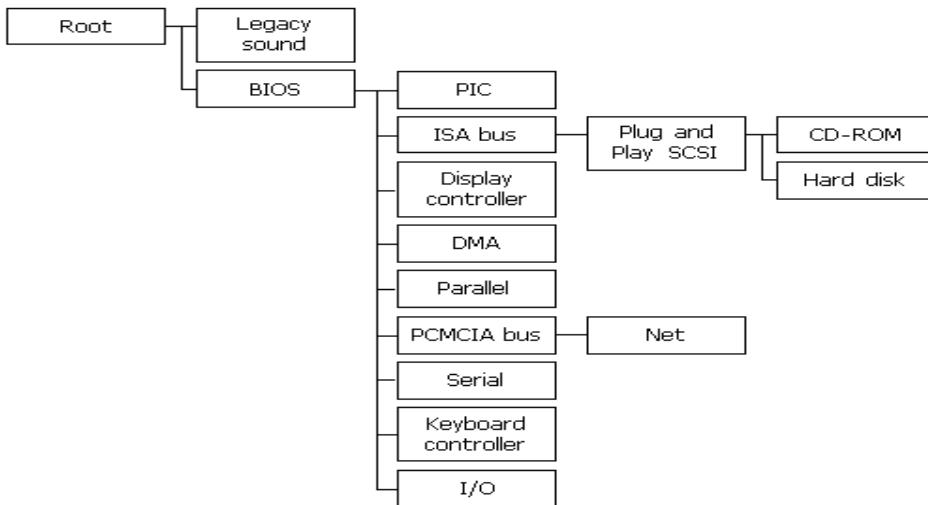
La arquitectura de Windows 98 incluye un componente llamado Gestor de Configuración, que dirige el proceso de configuración. Este proceso involucraría a muchas arquitecturas de bus y dispositivo coexistiendo en un único sistema, con más de un tipo de dispositivo usando la misma arquitectura de bus, aún cuando cada dispositivo tuviera unos requerimientos de configuración específicos.

El Gestor de Configuración trabaja con varios subcomponentes para identificar cada bus y cada dispositivo del sistema y para identificar la configuración establecida para cada dispositivo. El Gestor de Configuración se asegura de que cada dispositivo del ordenador pueda usar una petición de interrupción (IRQ), una dirección de puerto de E/S, y otros recursos sin entrar en conflicto con otros dispositivos.

Para realizar sus funciones, el Gestor de Configuración (realizado como una parte del VMM32) dispone de los enumeradores de bus para identificar todos los dispositivos en sus buses específicos y sus respectivas necesidades de recursos.

Los enumeradores de bus son drivers nuevos que son responsables de la creación del árbol de hardware de Windows 98. Un árbol de hardware es una representación jerárquica de todos los buses y dispositivos del ordenador. Cada bus y cada dispositivo es presentado como un nodo.

Gráfica 92. Ejemplo de árbol de Hardware de Windows 98



Durante el proceso de enumeración del dispositivo, el enumerador de bus localiza y reúne información de bien de los drivers de dispositivo o bien de los servicios de la BIOS para ese tipo particular de dispositivo. El Gestor de Configuración llama al árbitro de recursos para asignar los recursos a cada dispositivo.

Windows 98 proporciona árbitros para los recursos de E/S, la memoria, las interrupciones de hardware y los accesos directos a memoria (DMA).

Componentes del corazón del sistema

Se compone de tres componentes: Usuario, Kernel y GDI (Interface gráfica de dispositivos).

Cada uno de estos componentes incluye un par de librerías de enlace dinámico (DLL), una de 32 bits y otra de 16 bits, que proporcionan servicios para las aplicaciones que se ejecuten. Windows 98 está diseñado para usar código de 32 bit siempre que mejore el funcionamiento de la aplicación y no existan problemas de compatibilidad. Windows 98 mantiene el código de 16 bits donde se necesite para la compatibilidad o donde el uso del código de 32 bits incremente los requerimientos de memoria sin mejorar significantemente el funcionamiento.

Todos los subsistemas de E/S de Windows 98 (tales como las redes y el sistema de ficheros) y los driver de los dispositivos son de 32 bits, igual que todos los gestores de memoria y los componentes de planificación, incluyendo el Kernel y VMM.

1.3 Windows 2000

1.3.1 Ediciones de Windows 2000

- Windows 2000 profesional
- Windows 2000 Server
- Windows 2000 Advanced Server, y
- Windows 2000 Datacenter Server.

Se diferencian en el número de procesadores y memoria física soportados, el número de conexiones de red, y servicios (RAID, AD, DHCP, DNS, DFS, etc.)

1.3.2 Objetivos de diseño

Extensibilidad. Debe poder crecer y cambiar confortablemente según las necesidades de mercado.

Transportabilidad. El sistema debe ser capaz de ejecutarse sobre múltiples plataformas hardware y debe moverse a aquellas que aparezcan. Incluídos multiprocesadores.

Fiabilidad y robustez. El sistema debe protegerse él mismo de malfuncionamientos internos y ataques externos. Las aplicaciones no deben poder dañar al sistema o a otras aplicaciones.

Compatibilidad. Su API debe ser compatible con viejas versiones de Windows y MS-DOS. Debería poder interoperar con Unix, OS/2 y Netware.

Rendimiento. Dentro de las restricciones de los otros objetivos de diseño, el sistema debe ser tan rápido y responsive como sea posible sobre cada plataforma.

1.3.3 Modelos usados en implementación

Cliente-servidor. Ciertos servicios del SO están implementados como procesos de usuario.

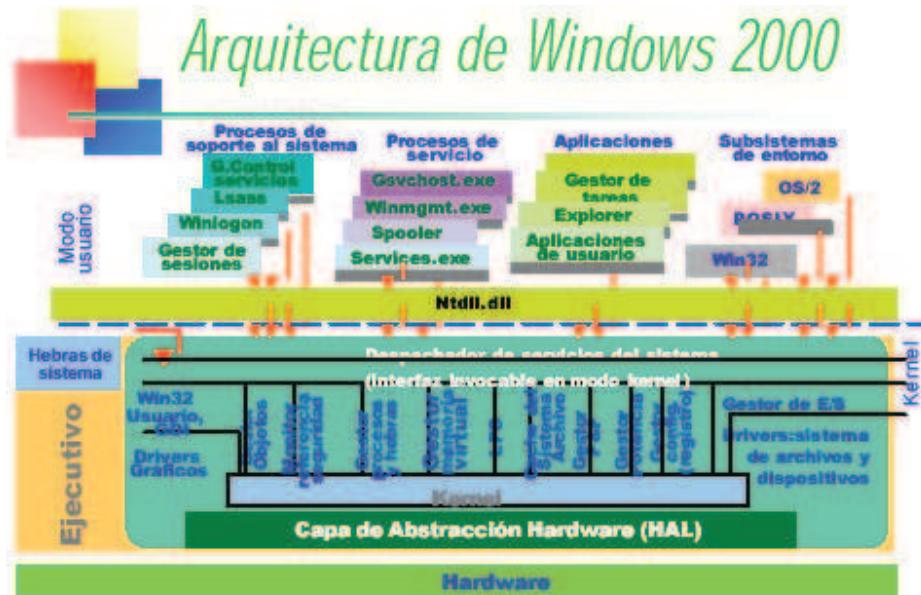
De capas. En el Ejecutivo, y la gestión de E/S.

Orientado a objetos. Si bien no es un SOOO, trata algunos elementos como objetos: procesos, hebras, archivos, puertos, semáforos, mutantes, etc.

Estructura microkernel. No un modelo puro, por razones de eficiencia.

Multiprocesamiento simétrico. Tanto la ejecución del SO como de varias hebras de una tarea se puede realizar en diferentes procesadores.

1.3.4 Arquitectura



1. El Ejecutivo (Executive)

El Ejecutivo contiene los componentes sensibles en rendimiento que se ejecutan en modo kernel, donde pueden interaccionar con el hardware, y entre ellos, sin incurrir en la sobrecarga de cambios de contexto y cambios de modo.

El programa NTOSKRNL.EXE contiene todos los componentes del Ejecutivo (excepto el subsistema Win32 -que está en WIN32K.SYS), y el kernel. Se carga en la mitad superior de memoria en la fase de arranque del sistema.

Funciones del ejecutivo

Exportadas y llamadas desde modo usuario (**servicios del sistema**). Exportadas por ntdll.dll.

Llamadas desde modo kernel pero exportadas y documentadas por Windows 2000 DDK (Device Driver Kit) o IFS Kit (Installable File System).

Subsistemas del Ejecutivo

Gestor de objetos. Crea, gestiona, y borra objetos del ejecutivo y TAD que representan recursos: procesos, hebras, etc.

Monitor de referencia de seguridad. Asegura la política de seguridad del computador local. Guarda los recursos realizando protección y auditoría en tiempo de ejecución.

Gestor de memoria virtual. Implementa memoria virtual, y soporte de base para el gestor de caché.

Gestor de E/S. Implementa E/S independientes del dispositivo.

Gestor de caché. Implementa una caché global de archivos.

Mecanismo LPC. Suministra paso de mensajes entre clientes y servidores de la misma máquina (una RPC eficiente y optimizada)

Gestor de configuraciones. Responsable de implementar y gestiona el Registro del sistema.

Gestor de procesos y hebras. Exporta procesos y hebras a través de la API.

Win32 Implementa funciones de mensajes y dibujo de Win32.

Gestor de Potencia. Controla el estado de potencia de la máquina.

Gestor de Plug-and-Play. Notifica a los gestores de dispositivos la puesta en línea o desconexión de los dispositivos.

Despachador de servicios del sistema

El Kernel. El Kernel contiene:

- Planificador de hebras.
- Primitivas de sincronización (mutex, semáforos, eventos y cerrojos espera ocupada (*spin locks*))
- Despacho de interrupciones y excepciones.
- Rutinas y objetos base usados por el Ejecutivo.
- Este módulo nunca es paginado fuera de memoria y su ejecución nunca es apropiada por una hebra.

Capa Abstracción Hardware (HAL)

La HAL (archivo Hall.dll) aísla al kernel, manejadores de dispositivos y al resto del ejecutivo de las diferencias hardware de la plataforma.

HAL exporta un modelo de procesador común. Los gestores de dispositivos utilizan este procesador común más que una CPU específica. La existencia de diferentes placas para el mismo procesador (por ejemplo multiprocesador) base se asegura mediante la escritura de una HAL personalizada por el propio fabricante.

Sistema gráfico y de ventanas

El sistema gráfico y de ventanas implementa las funciones de la interfaz gráfica de usuario (GUI), tales como las que tratan con ventanas, control de la interfaz de usuario, y dibujo.

Implementado por los archivos Win32.dll, Kernel32.dll, Advapi32.dll, User32.dll y Gdi32.dll.

Este sistema estaba en la versión de NT 3.5 en modo usuario, y de introdujo en NT 4.0 en modo kernel por razones de eficiencia.

Procesos de usuario

Procesos de soporte del sistema, tales como *logon* y el *gestor de sesiones*, que no son servicios Win 2000 (no iniciados por el gestor de control de servicios).

Procesos servidores, son servicios 2000 (equivalentes a demonios UNIX) tales como el *Event Logger*, *Microsoft SQL Server* y *Microsoft Exchange Server*.

Subsistemas de entorno, que exponen los servicios nativos del SO a las aplicaciones de usuario y suministran un entorno de SO: Win32, Posix, y OS/2.

Aplicaciones de usuario, que pueden ser de los tipos: Win32, Windows 3.1, MS-DOS, Posix, ó OS/2 1.2.

Subsistemas: entorno y DLLs

Un ejecutable se liga en tiempo de enlace a un único subsistema. Al ejecutarse, se examina el tipo de subsistema (cabecera de la imagen) para notificar al correspondiente subsistema de la creación del proceso. Por ejemplo un programa Win32 se enlaza con las DLLs del lado-cliente de Win32 (kernel32.dll, gdi32.dll, y user32.dll); uno POSIX lo hace con psxdll.dll, etc.

La aplicaciones no invocan llamadas nativas directamente; en su lugar, utilizan las DLLs de los subsistemas para traducir una función documentada en la no documentada apropiada nativa de Windows 2000.

Entornos de sistema operativo

La DLL cliente se ejecuta en beneficio de su servidor.

Ofrece una API completa, o invoca al servidor (modificar información del entorno). DLLs y servidor pueden usar los servicios nativos. La API de entorno de SO aumenta la funcionalidad o semántica que le son específicas.

Servicios de sistema nativos

La API nativa es similar a la API de Win32, y no da a la aplicación más privilegios que el subsistema de entorno.

Las aplicaciones y subsistemas de entorno acceden a la API nativa a través de la ntdll.dll, (contiene entre otras cosas los puntos de entrada a los servicios del sistema).

Existen más de 200 llamadas nativas.

Manejadores de dispositivos

Los manejadores de dispositivos son módulos en modo kernel cargables que hace de interfaz entre el Gestor de E/S y el hardware.

Se ejecutan en modo kernel en uno de los tres contextos siguientes:

- Contexto de la hebra que inicio la E/S
- Contexto de una hebra de sistema en modo kernel.
- Como resultado de una interrupción, en cuyo caso no utiliza el contexto de ninguna hebra.

Tipos de manejadores

Hardware. Manipulan el hardware para realizar E/S, por ejemplo driver de bus, de disco, etc.

Sistemas de archivos. Aceptan solicitudes de E/S orientadas a archivos y las traducen a solicitudes E/S ligadas al correspondiente dispositivo.

Filtros de sistemas de archivos. Interceptan E/S y le dan un valor añadido antes de pasarla a la capa siguiente, por ejemplo mirroring, encriptación.

Redireccionadores de red. Manejadores de sistemas de archivos que transmiten/reciben operaciones de E/S de sistemas de archivos a otra máquina.

Protocolos. Implementan los protocolos de red, tales como TCP/IP, NetBUI, e IPX/SPX.

Filtros de flujo kernel. Se encadenan para realizar procesamiento de señales sobre flujos de datos, tales como grabar y reproducir vídeo/audio.

Procesos de sistema

Proceso ocioso (hay uno por CPU).

Proceso de sistema. Contiene gran parte de hebras de sistema en modo kernel, por ejemplo, la hebra *Balance Set Manager*, control de dispositivos.

Gestor de sesiones (smss.exe). Primer proceso en modo usuario; gestiona parte de la inicialización del sistema, y lanza csrss, y winlogon.

Subsistema Win32 (csrss.exe).

Proceso Logon (winlogon.exe). Maneja las entradas/salidas de usuarios interactivos. Se activa con la *secure attention sequence* (SAS).

1.4 Windows Server 2003

1.4.1 Características

- Disponible en cuatro versiones: Standard, Enterprise, Datacenter y Web.
- Enterprise y Datacenter cuentan con versiones para plataformas de 64 bits (Itanium) y con configuraciones de hasta 8 nodos en *cluster*, mientras que

la versión para entornos de Internet se puede comprar sin las licencias de todos aquellos servicios ajenos al propio de servidor web.

- Soporta configuraciones en cluster de hasta 8 nodos.
- Mejoras en cuanto a rendimiento y eficiencia.
- Windows Server 2003 es una infraestructura integrada y segura diseñada para ayudar a los clientes a reducir costes e incrementar la eficacia de las operaciones IT (Itanium).
- La nueva plataforma ayuda a los clientes a ampliar los recursos existentes mientras sientan las bases para construir una nueva generación de aplicaciones conectadas que mejoran la productividad del negocio.
- Esta familia de servidores soporta también la versión de 64 bit de Microsoft Windows Server 2003. Configurado con los procesadores 64 basados en Itanium con 512 GB de memoria sobre Datacenter Edition de Windows .NET.

1.4.2 Arquitectura

Abandonó la estructura relativamente monolítica de anteriores versiones de IIS, donde un solo componente manejaba las peticiones HTTP y los procesos de administración. En su lugar, se ha construido un servicio formado por varios componentes especializados, que optimizan cada tarea.

La innovación más destacable en cuanto a rendimiento se refiere, es que se ha incluido en el núcleo del sistema operativo un componente llamado HTTP.SYS, dedicado a manejar las peticiones HTTP.

Gráfica 93. Estructura del sistema operativo Windows 2003 Server
Fuente: www.windowstimag.com



1.4.3 Descripción

Fácil de implementar, administrar y usar

Windows Server 2003 es fácil de usar. Los nuevos asistentes simplificados facilitan la configuración de funciones específicas de servidor y de las tareas habituales de administración de servidores, de tal forma que incluso los servidores que no disponen de un administrador dedicado son fáciles de administrar. Además, los administradores disponen de diversas funciones nuevas y mejoradas, diseñadas para facilitar la implementación de Active Directory. Las réplicas de Active Directory de gran tamaño pueden implementarse desde medios de copia de seguridad, y la actualización desde sistemas operativos de servidor anteriores, como Microsoft Windows NT, es más fácil gracias a la Herramienta de Migración de Active Directory (ADMT), que copia contraseñas y permite la creación de secuencias de comandos.

El mantenimiento de Active Directory es más fácil con las funciones nuevas, como la posibilidad de cambiar el nombre de los dominios y de volver a definir esquemas, proporcionando a los administradores la flexibilidad necesaria para controlar los cambios organizativos que puedan producirse. Además, las relaciones de confianza entre bosques permiten a los administradores conectar los bosques de Active Directory, proporcionando autonomía sin sacrificar la integración.

Las herramientas de implementación mejoradas, como los Servicios de instalación remota, ayudan a los administradores a crear rápidamente imágenes del sistema y a implementar servidores.

Infraestructura segura

La administración de identidades en Active Directory abarca la totalidad de la red. El cifrado de datos confidenciales resulta sencillo, y las directivas de restricción de software pueden usarse para prevenir los daños causados por virus y otro tipo de código malintencionado. Windows Server 2003 es adecuado para implementar una infraestructura de claves públicas (PKI), tiene funciones de inscripción automática y de renovación automática.

Confiabilidad, disponibilidad, escalabilidad y rendimiento de nivel empresarial

Se ha mejorado la confiabilidad mediante una gama de funciones nuevas y mejoradas, como el reflejo de memoria, la memoria agregada en caliente y la detección de estado en Internet Information Services (IIS) 6.0. Para tener una mayor disponibilidad, el servicio Microsoft Cluster admite ahora clústeres de hasta ocho nodos y nodos separados geográficamente. Se proporciona una mayor

escalabilidad, con la posibilidad de escalar desde un único procesador hasta sistemas de 32 direcciones. Globalmente, Windows Server 2003 es más rápido, con un rendimiento del sistema de archivos hasta un 140 por ciento superior, así como un rendimiento significativamente más rápido para Active Directory, los servicios Web XML, los Servicios de Terminal Server y las redes.

Creación fácil de sitios Web de Internet e intranet dinámicos

IIS 6.0, el servidor Web incluido en Windows Server 2003, proporciona una seguridad avanzada y una arquitectura confiable que ofrece aislamiento para las aplicaciones y un rendimiento muy mejorado.

Cambios en el diseño de IIS

El IIS 6 ahora proporciona:

- **Soporte de tecnología .NET.** Las aplicaciones basadas en ASP.NET, que se ejecutan bajo IIS, son la clave para la construcción de aplicaciones Web y Web Services. Para soportar con garantías estas aplicaciones, IIS ha tenido que hacerse más fiable y seguro, y ser capaz de manejar un gran número de peticiones.
- **Demandas de clientes.** Hay una serie de características que se deben cubrir para satisfacer la demanda como: estabilidad suficiente para que el funcionamiento 24x7 sea normal y no excepcional, facilidad de manejo más centralizado, soportar miles de sitios Web por servidor hardware, escalabilidad tanto vertical como horizontal.
- **Competencia.** Los servidores Web del tipo “*open source*” tales como Apache sobre plataforma Linux o FreeBSD, ofrecen unas características muy atractivas a muy bajo coste.

Para satisfacer estos requerimientos, Microsoft ha dirigido sus esfuerzos en mejorar IIS en los siguientes aspectos:

- **Rendimiento.** Se ha trasladado al núcleo el código que procesa el *Hypertext Transfer Protocol* (HTTP) aumentando la capacidad de manejar peticiones, y se ha mejorado la capacidad de las aplicaciones para aprovechar los servidores multiprocesador y las granjas de servidores. Estos cambios pueden acelerar la entrega de contenido estático y la ejecución de páginas dinámicas, compitiendo con otros productos “*open source*”.
- **Fiabilidad y tolerancia a fallos.** Microsoft ha reestructurado el modelo de procesos para proteger de manera más efectiva a IIS de código errante y proteger los sitios Web unos de otros. Estos cambios son cruciales para proporcionar un alto nivel de disponibilidad a sitios de Web de comercio electrónico y Web Services.

- **Seguridad.** El código de IIS se ha visto sujeto a una revisión de seguridad y se han hecho cambios radicales en su configuración inicial. Estos cambios podrían hacerlo más resistente a vulnerabilidades de seguridad.

Desarrollo rápido con el servidor de aplicaciones integrado

Microsoft .NET Framework está profundamente integrado en el sistema operativo Windows Server 2003. Microsoft ASP.NET permite la creación de aplicaciones Web de alto rendimiento. Con la tecnología conectada a .NET, los desarrolladores ya no están obligados a crear código pesado y estático, y pueden trabajar eficazmente con los lenguajes de programación y las herramientas que ya conocen. Las aplicaciones ya existentes pueden volver a empaquetarse fácilmente como servicios Web XML. Las aplicaciones UNIX pueden integrarse o migrarse fácilmente. Y los desarrolladores pueden crear rápidamente aplicaciones Web preparadas para servicios y dispositivos móviles mediante los controles de formularios Web móviles de ASP.NET y otras herramientas.

Herramientas de administración sólidas

La nueva consola de administración de directivas de grupo (GPMC) permite a los administradores implementar y administrar mejor las directivas que automatizan las áreas de configuración de claves, como los perfiles móviles, la seguridad, la configuración y los escritorios de los usuarios.

1.4.4 Versiones

La familia de servidores **Windows 2003** está formada por cuatro versiones:

- **Web Edition**
- **Standard Edition**
- **Enterprise Edition**
- **Datacenter Edition**

Cada una de ellas necesita los siguientes requisitos del sistema:

Requisito	Web Edition	Standard Edition	Enterprise Edition	Datacenter Edition
Velocidad mínima de la CPU	133 MHz	133 MHz	133MHz para equipos basados en x 86. 733 MHz para equipos basados en Itanium.	133MHz para equipos basados en x 86. 733 MHz para equipos basados en Itanium
Velocidad recomendada de la CPU	550 MHz	550 MHz	733 MHz	733 MHz

Requisito	Web Edition	Standard Edition	Enterprise Edition	Datacenter Edition
Memoria RAM mínima	128 MB	128 MB	128 MB	512 MB
Memoria RAM mínima recomendada	256 MB	256 MB	256 MB	1 GB
Memoria RAM máxima	2 GB	4GB	32 GB para equipos basados en x86. 64 GB para equipos basados en Itanium.	64 GB para equipos basados en x86. 128 GB para equipos basados en Itanium.
Soporte para multiprocesadores	1 o 2	Hasta 4	Hasta 8	Un mínimo de 8. Un máximo de 32 para equipos basados en x86. Un máximo de 64 para equipos basados en Itanium.
Espacio en disco necesario para la instalación	1.5 GB	1.5 GB	1.5 GB para equipos basados en X86. 2.0 GB para equipos basados en Itanium.	1.5 GB para equipos basados en x86. 2.0 GB para equipos basados en Itanium.

1.5 WINDOWS XP

1.5.1 Historia

Windows XP es una continuación de Windows 2000 en el hecho de que admite la especificación de la interfaz de configuración y energía (ACPI), que proporciona una administración de energía y una configuración del sistema seguras.

XP, está construido sobre los modernos sistemas operativos Windows NT y, los que, hasta ahora, eran la gama alta de Microsoft destinada a empresas y servidores. Windows XP fue construido sobre el motor renovado de Windows por lo que, en teoría, debe ser igual de estable.

1.5.2 Características

Windows XP viene en dos versiones que son Windows XP professional y Windows XP Home Edition.

Windows XP requiere al menos 128 MB de RAM, 1.5 GB de disco duro (algo más si se quiere instalar después otras cosas aparte del sistema operativo) y un procesador de 233 MHz. Y esto sólo es el mínimo. Para trabajar en condiciones óptimas no servirá casi ningún ordenador con más de un año y medio de antigüedad.

Windows XP ofrece características de administración de energía mejoradas para los equipos móviles y de escritorio. También proporciona una compatibilidad más

limitada para la administración de energía de sistemas basados en la API de administración de energía avanzada.

Es un sistema compatible con ACPI, el sistema operativo administra, dirige y coordina la energía de modo que el sistema esté accesible para los usuarios al instante, cuando sea necesario, mientras que se mantienen en silencio y con el menor consumo de energía posible cuando no está funcionando activamente.

Windows XP incluye las siguientes características para la administración de energía:

- Funcionamiento mejorado de reinicio y reanudación. En Windows XP se reducen los intervalos de espera de inicio y cierre de modo que el equipo esté preparado para utilizarlo rápidamente cuando el usuario lo active.
- Energía eficaz. Las características que mejoran la eficacia de la energía, especialmente para los equipos portátiles, incluyen la compatibilidad nativa de tecnologías de control de rendimiento del procesador, la atenuación de la pantalla LCD cuando se utilice la energía de la batería y la desactivación del panel de la pantalla del equipo móvil cuando se cierre la tapa.
- Compatibilidad de activación. El equipo aparentemente está apagado cuando no se utiliza, pero responde a sucesos de activación, como una llamada de teléfono o una solicitud de red.
- Interfaz de usuario para establecer preferencias de energía. Opciones de energía , en el panel de control , proporciona una interfaz de usuario con la que se pueden establecer preferencias mediante la selección o creación de esquemas de energía, la especificación de opciones de utilización de batería y el establecimiento de alarmas de baja energía. Si hay un sistema de energía ininterrumpida (UPS), Opciones de energía administrativa también dicho sistema.
- Directiva de energía independiente para cada dispositivo. Todos los dispositivos diseñados para que utilicen las características de administración de energía de windows pueden participar en la administración de la energía. Cuando no se están utilizando, pueden solicitar que el sistema operativo los coloque en un estado de bajo consumo para conservar la energía.

1.5.3 Windows XP Professional

Está basada en el código central de software utilizado en Windows 2000 y Windows NT workstation. Este código, conocido como núcleo de windows NT, o el nuevo motor de Windows, hace que Windows XP sea más eficaz, seguro y estable que

Windows Me, Windows 98 o Windows 95. Incluso cuando se bloquea un programa, el sistema seguirá ejecutándose.

Seguridad del protocolo Internet (IPSec). El protocolo estándar que se utiliza para enviar y recibir información a través de Internet hace que los datos sean susceptibles de ser interceptados, modificados, reproducidos o falsificados. Sin seguridad, tanto las redes públicas como privadas pueden ser objeto de vigilancia y acceso no autorizado.

Windows XP Professional utiliza IPSec para ofrecer una plataforma ideal para proteger las comunicaciones en una intranet o en Internet.

Antes de enviar los datos, IPsec negocia con el equipo al que está conectado. IPsec establece un nivel de seguridad apropiado para la sesión de comunicación. Despues, genera una clave de autentificación secreta y autentica la identidad del otro equipo, antes de comenzar a intercambiar los datos de manera segura.

Escenarios de reinicio reducidos drásticamente. Elimina la mayoría de los escenarios que obligaban a los usuarios finales a reiniciar los equipos en Windows NT 4.0 y Windows 95/98/Me. Además, una gran parte de las instalaciones de software no requieren reiniciar.

Protección de códigos mejorada. Las estructuras de los datos importantes del núcleo son de sólo lectura, por lo que los controladores y las aplicaciones no pueden corromperlas. Todos los códigos de controladores de dispositivos son de sólo lectura y con protección de página.

Soporte. Proporciona un mecanismo para instalar y ejecutar colateralmente varias versiones de componentes individuales de Windows. Esto ayuda a resolver el problema “DLL hell”, al permitir que una aplicación escrita y probada con una versión de un componente del sistema siga utilizando la misma versión, aunque se instale una aplicación que utilice una versión más reciente del mismo componente.

Protección de archivos de Windows. Protege los archivos principales del sistema contra la sobrescritura por la instalación de aplicaciones. Si se sobrescribe un archivo, la protección de archivos de Windows restaura la versión correcta.

Instalador de Windows. Servicio del sistema que ayuda al usuario a instalar, configurar, realizar el seguimiento y quitar programas de software correctamente. Ayuda a minimizar los períodos de inactividad y aumenta la estabilidad del sistema.

Arquitectura multitarea preferente. Su diseño permite que varias aplicaciones se ejecuten simultáneamente, al tiempo que garantiza una gran respuesta y estabilidad del sistema. **Memoria escalable y soporte de procesador** Admite

hasta 4 gigabytes (GB) de memoria RAM y hasta dos multiprocesadores simétricos.

Sistema de cifrado de archivos (EFS) con soporte para varios usuarios. Cifra todos los archivos con una clave generada aleatoriamente. Los procesos de cifrado y descifrado son transparentes para el usuario. En Windows XP Professional, EFS permite que varios usuarios tengan acceso a un documento cifrado. El más alto nivel de protección contra piratas informáticos y robo de datos.

Seguridad IP (IPSec). Ayuda a proteger los datos transmitidos a través de una red. IPSec es una parte importante de la seguridad de las redes virtuales privadas (VPN), que permiten a las organizaciones transmitir datos de forma segura a través de Internet. Los administradores de tecnologías de la información podrán crear redes virtuales privadas seguras con rapidez y facilidad.

Soporte para Kerberos. Proporciona el estándar industrial y autenticación de alto nivel con un único inicio de sesión para los recursos de la empresa basados en Windows 2000. Kerberos es un estándar de Internet especialmente eficaz en redes que incluyen sistemas operativos diferentes, como UNIX.

Soporte para tarjetas inteligentes. Las capacidades de tarjeta inteligente están integradas en el sistema operativo, incluido el soporte para el inicio de sesión con tarjetas inteligentes en sesiones del servidor de terminal alojadas en servidores basados en Windows .Server 2003 (la plataforma de servidores de la próxima generación). Las tarjetas inteligentes mejoran las soluciones sólo para software, como la autenticación de clientes, el inicio de sesión interactivo, la firma de código y el correo electrónico seguro.

Hibernación. Tras un tiempo establecido, o cuando se requiera, Windows XP Professional guarda la memoria en el disco y desconecta la energía. Cuando se restaura la energía, todas las aplicaciones se vuelven a abrir exactamente como se dejaron.

Fiabilidad a nivel empresarial. Windows XP proporciona un nuevo nivel de estabilidad, para que usted se pueda concentrar en el trabajo. Por ejemplo, en la mayoría de los casos, si se produce un error en un programa, el equipo seguirá funcionando.

Rendimiento avanzado. Windows XP administra los recursos del sistema con eficacia; obtiene los mismos niveles de rendimiento que Windows 2000 y sobrepasa en un 46% los de Windows 98 Segunda edición.

Escritorio remoto. Escritorio remoto permite crear una sesión virtual y utilizar el equipo de escritorio desde otro equipo que ejecute Windows 95 o posterior, lo que

le permitirá tener acceso a todos los datos y aplicaciones aunque no se encuentre en la oficina.

Nuevo diseño visual basado en tareas. Por medio de un diseño más claro y nuevas pistas visuales llegará rápidamente a las tareas que más utiliza.

Soporte para redes inalámbricas 802.1x. El soporte para redes inalámbricas 802.1x proporciona soporte para acceso seguro, además de mejoras de rendimiento en redes inalámbricas.

Windows Messenger. Windows Messenger es la manera más fácil de comunicarse y colaborar en tiempo real utilizando el equipo. Puede ver el estado en línea de sus personas de contacto y elegir comunicarse con ellas mediante texto, voz o vídeo con un mejor rendimiento y una calidad superior.

Sistema de codificación de archivos. El sistema de codificación de archivos proporciona un alto nivel de protección contra piratas informáticos y el robo de datos, mediante la codificación transparente de los archivos con una clave generada aleatoriamente.

Rápida reanudación desde el modo de hibernación o suspensión. Ahorre la energía de las baterías cuando trabaje fuera de la oficina. Con Windows XP su equipo portátil puede entrar en modo de suspensión o hibernación y comenzar a trabajar rápidamente tras reanudar la actividad desde la suspensión o hibernación.

Centro de ayuda y soporte con asistencia remota. Además de una exhaustiva documentación, el Centro de ayuda y soporte de Windows XP incluye Asistencia remota, que le permite que un amigo o un profesional de tecnologías de la información, que también ejecute Windows XP, controle de manera remota su equipo para mostrarle un proceso o ayudarle a resolver un problema.

Restaurar sistema. Si se produce algún error en el equipo, puede devolver el sistema a su estado anterior.

1.5.4 Windows XP Home Edition

El sistema operativo Windows XP Home Edition ofrece un conjunto de nuevas características que ayudan a trabajar de manera más inteligente, conectarse más rápido a Internet y con otros usuarios.

Protección de archivos de Windows. Impide que las aplicaciones cambien accidentalmente los archivos importantes del sistema operativo. De esta manera, el sistema se protege de forma activa y automática.

Arquitectura de modo de núcleo protegido. Las aplicaciones no tienen acceso al núcleo del código de software en que se basa el sistema operativo. De esta manera, aumenta significativamente la fiabilidad del sistema.

Separación de procesos. Las aplicaciones de roaming no provocan daños en el equipo. Cada aplicación está en un espacio de memoria completamente separado y protegido.

Monitor del sistema. Analiza automáticamente cientos de diferentes medidas del sistema, por ejemplo, memoria, disco y rendimiento de la red.

Administrador de tareas. Ofrece información útil acerca del rendimiento del equipo y permite terminar programas inactivos. También puede solicitar un mecanismo para informes opcional que se ajuste mejor a sus requisitos específicos.

Redes domésticas. Permiten:

- Configurar una red doméstica, incluidas las conexiones físicas para impresoras o faxes, e instalar protocolos y puentes.
- Compartir una conexión de Internet con todos los equipos de una red.
- Compartir recursos en un equipo.

Novedades:

- La red doméstica se ha mejorado con el servidor de seguridad de conexión a Internet para ayudarle a protegerse de accesos no autorizados mientras está conectado a Internet.
- Conexión compartida a Internet. Permite que varios equipos domésticos tengan acceso a Internet al mismo tiempo a través de la misma conexión de banda ancha o de acceso telefónico.
- Windows XP Home Edition ya dispone de una opción para desconectar de forma remota la conexión de acceso telefónico y poder utilizar la línea de teléfono y volver a conectarse de nuevo fácilmente.
- Asistente para la publicación en la red. El Asistente para la publicación en Web muestra cómo publicar imágenes en Internet fácil y rápidamente para que pueda compartirlas con otros.

Windows XP Professional le proporciona todas las ventajas de Windows XP Home Edition, además de características adicionales de acceso remoto, seguridad, rendimiento, facilidad de uso y soporte multilingüe.

A continuación se presenta un paralelo con las características correspondientes a las dos versiones de XP, vistas anteriormente:

Características	Windows XP Home Edition	Windows XP Professional
Todas las características de Windows XP Home Edition		
Nueva interfaz de usuario: encontrará lo que necesita cuando lo necesite con más facilidad.	SI	SI
Una base en la que puede confiar: mantiene su equipo funcionando cuando más lo necesita.		
Reproductor de Windows Media para Windows XP: un solo lugar para buscar, reproducir, organizar y almacenar medios digitales.		
Asistente para configuración de red: conecte y comparta fácilmente los equipos y dispositivos de su hogar.		
Windows Messenger: lo último en herramientas de comunicaciones y colaboración, con mensajería instantánea, conferencias de vídeo y voz y aplicaciones compartidas.		
Centro de ayuda y soporte técnico: permite una fácil resolución de los problemas y obtener ayuda y soporte técnico cuando lo necesite.		
Soporte para dispositivos móviles: acceso a la información mientras se encuentra de viaje		
Compatibilidad avanzada con equipos portátiles (incluida la compatibilidad con ClearType, DualView, mejoras en la administración de la energía): le permite realizar el mismo trabajo mientras está de viaje que en la oficina.	SI	SI
Conexiones inalámbricas: configuración de red inalámbrica 802.1x automática.	SI	SI
Escritorio remoto: obtenga acceso remoto a su equipo con Windows XP Professional desde otro equipo con Windows, de manera que pueda trabajar con todos sus datos y aplicaciones mientras esté fuera de la oficina.	NO	SI
Archivos y carpetas sin conexión: tenga acceso a archivos y carpetas de un recurso compartido de red cuando esté desconectado del servidor.	NO	SI
Alta disponibilidad con capacidad para trabajar en varias tareas a la vez		
Inicio rápido y mejoras en la administración de la energía: tiempos de reinicio y reanudación más rápidos.	SI	SI
Multitarea: permite ejecutar varias aplicaciones a la vez.	SI	SI
Compatibilidad con procesador escalable: admite varios procesadores bidireccionales.	NO	SI
Mantiene sus datos y su intimidad a salvo		
Servidor de seguridad de conexión a Internet: protege automáticamente el equipo contra el acceso no autorizado cuando esté conectado a Internet.	SI	SI
Protección de la confidencialidad de Internet Explorer 6: mantiene el control sobre su información personal cuando visite sitios Web.	SI	SI
Sistema de cifrado de archivos: protege los datos confidenciales de los archivos que se almacenan en el disco mediante el sistema de archivos NTFS.	NO	SI
Control de acceso: restringe el acceso a los archivos, las aplicaciones y otros recursos seleccionados.	NO	SI
Diseñado para funcionar con servidores y soluciones de administración de Microsoft Windows		

Administración centralizada: combine servidores Windows XP Professional con un dominio de un servidor Windows para aprovechar las ventajas de la completa gama de eficaces herramientas de administración y seguridad.	NO	SI
Directiva de grupo: simplifica la administración de grupos de usuarios y equipos.	NO	SI
Instalación y mantenimiento de software: instale, configure, repare y quite aplicaciones automáticamente.	NO	SI
Perfiles móviles de usuario: obtenga acceso a todos sus documentos y valores de configuración sin importar dónde inicia la sesión.	NO	SI
Servicio de instalación remota (RIS): permite la instalación remota de sistemas operativos donde los escritorios se pueden instalar a través de la red.	NO	SI
Comuníquese con personas de todo el mundo de un modo eficaz		
Modo binario único en todo el mundo: escriba texto en cualquier idioma y ejecute la versión en cualquier idioma de aplicaciones Win32 en cualquier versión de Windows XP.	SI	SI
Complemento Interfaz de usuario multilingüe (MUI): cambie el idioma de la interfaz de usuario para obtener cuadros de diálogo, menús, archivos de ayuda, diccionarios, herramientas de corrección, etc. en otros idiomas.	NO	SI

1.6 Windows NT

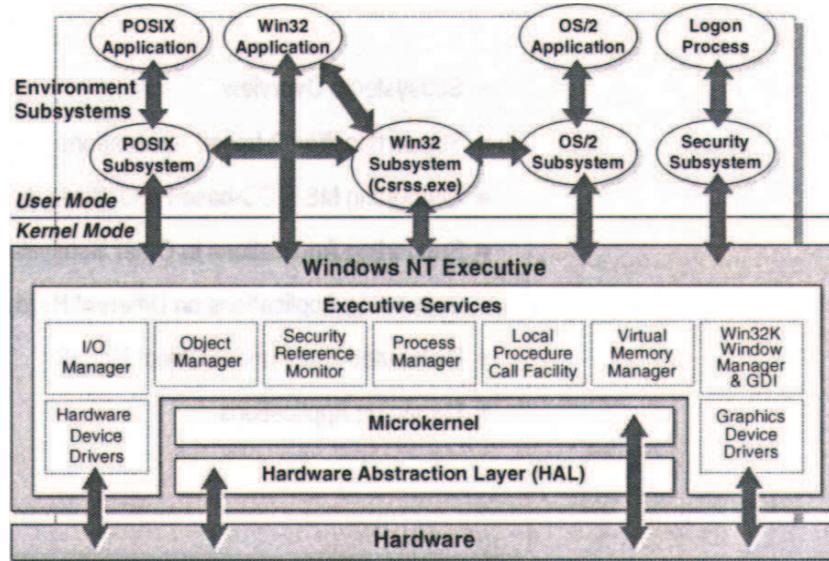
1.6.1 Arquitectura

Windows NT presenta una arquitectura del tipo cliente-servidor. Los programas de aplicación son contemplados por el sistema operativo como si fueran clientes a los que hay que servir, y para lo cual viene equipado con distintas entidades servidoras uno de los objetivos fundamentales de diseño fue el tener un núcleo tan pequeño como fuera posible, en el que estuvieran integrados módulos que dieran respuesta a aquellas llamadas al sistema que necesariamente se tuvieran que ejecutar en modo privilegiado (también llamado modo kernel, modo núcleo y modo supervisor). El resto de las llamadas se expulsarían del núcleo hacia otras entidades que se ejecutarían en modo no privilegiado (modo usuario), y de esta manera el núcleo resultaría una base compacta, robusta y estable. Por eso se dice que Windows NT es un sistema operativo basado en micro-kernel

En la arquitectura se distingue un núcleo que se ejecuta en modo privilegiado, y se denomina Executive, y unos módulos que se ejecutan en modo no privilegiado, llamados subsistemas protegidos.

Los programas de usuario (también llamados programas de aplicación) interaccionan con cualquier sistema operativo (S.O. en adelante) a través de un juego de llamadas al sistema propio de dicho sistema. En el mundo Windows en general, las llamadas al sistema se denominan API (Application Programming Interfaces, interfaces para la programación de aplicaciones).

Gráfica 94. Arquitectura Windows NT



El núcleo se ejecuta en modo privilegiado (Executive) y en modo no privilegiado (subsistemas protegidos).

Los subsistemas protegidos

Son una serie de procesos servidores que se ejecutan en modo no privilegiado, al igual que los procesos de usuario, pero que tienen algunas características propias que los hacen distintos.

Se inician al arrancar el S.O. y existen dos tipos: integrales y de entorno.

Un **subsistema integral** es aquel servidor que ejecuta una función crítica del S.O. (como por ejemplo el que gestiona la seguridad). Un **subsistema de entorno** da soporte a aplicaciones procedentes de S.O. distintos, adaptándolas para su ejecución bajo Windows NT. Existen tres de este tipo:

- Win32, que es el principal, y proporciona la interfaz para aplicaciones específicamente construidas para Windows NT.
- POSIX, que soporta aplicaciones UNIX.
- OS/2, que da el entorno a aplicaciones procedentes del S.O. del mismo nombre.

El subsistema Win32. Es el más importante, ya que atiende no sólo a las aplicaciones nativas de Windows NT, sino que para aquellos programas no Win32, reconoce su tipo y los lanza hacia el subsistema correspondiente. En el caso de

que la aplicación sea MS-DOS o Windows de 16 bits (Windows 3.11 e inferiores), lo que hace es crear un nuevo subsistema protegido. Así, la aplicación DOS o Win16 se ejecutaría en el contexto de un proceso llamado VDM (Virtual DOS Machine, máquina virtual DOS), que no es más que un simulador de un ordenador funcionando bajo MS-DOS. Las llamadas al API Win16 serían correspondidas con las homónimas en API Win32. Microsoft llama a esto WOW (Windows On Win32). El subsistema soporta una buena parte del API Win32. Así, se encarga de todo lo relacionado con la interfaz gráfica con el usuario (GUI), controlando las entradas del usuario y salidas de la aplicación.

El subsistema POSIX. La norma POSIX (Portable Operating System Interface for UNIX) fue elaborada por IEEE para conseguir la portabilidad de las aplicaciones entre distintos entornos Windows NT, UNIX, VMS, etc. Se trata de un conjunto de 23 normas, identificadas como IEEE 1003.0 a IEEE 1003.22, o también POSIX.0 a POSIX.22, de las cuales el subsistema POSIX soporta la POSIX.1, que define un conjunto de llamadas al sistema en lenguaje C. El subsistema sirve las llamadas interaccionando con el Executive.

El subsistema OS/2. Igual que el subsistema POSIX proporciona un entorno para aplicaciones UNIX, este subsistema da soporte a las aplicaciones del S.O. OS/2. Proporciona la interfaz gráfica y las llamadas al sistema; las llamadas son servidas con ayuda del Executive.

El subsistema proceso de inicio. El proceso de inicio (Logon Process) recibe las peticiones de conexión por parte de los usuarios. En realidad son dos procesos, cada uno encargándose de un tipo distinto de conexión: el proceso de inicio local, que gestiona la conexión de usuarios locales directamente a una máquina Windows NT; y el proceso de inicio remoto, el cual gestiona la conexión de usuarios remotos a procesos servidores de NT.

El subsistema de seguridad. Este subsistema interacciona con el proceso de inicio y el llamado monitor de referencias de seguridad, de esta forma se construye el modelo de seguridad en Windows NT. El subsistema de seguridad interacciona con el proceso de inicio, atendiendo las peticiones de acceso al sistema. Consta de dos subcomponentes: la autoridad de seguridad local y el administrador de cuentas.

El primero es el corazón del subsistema de seguridad, en general gestiona la política de seguridad local, así, se encarga de generar los permisos de acceso, de comprobar que el usuario que solicita conexión tiene acceso al sistema, de verificar todos los accesos sobre los objetos (para lo cual se ayuda del monitor de referencias a seguridad) y de controlar la política de auditorías, llevando la cuenta de los mensajes de auditoría generados por el monitor de referencias. El administrador de cuentas mantiene una base de datos con las cuentas de todos los usuarios (login, claves, identificaciones, etc.).

El Executive. No debemos confundir el Executive con el núcleo de Windows NT, aunque muchas veces se usan (incorrectamente) como sinónimos. El Executive consta de una serie de componentes software, que se ejecutan en modo privilegiado, uno de los cuales es el núcleo. Dichos componentes son totalmente independientes entre sí, y se comunican a través de interfaces bien definidas. En el diseño se procuró dejar el núcleo tan pequeño como fuera posible y, su funcionalidad es mínima.

El administrador de objetos (Object Manager). Se encarga de crear, destruir y gestionar todos los objetos del Executive. Se tiene infinidad de objetos: procesos, subprocessos, ficheros, segmentos de memoria compartida, semáforos, mutex, sucesos, etc. Los subsistemas de entorno (Win32, OS/2 y POSIX) también tienen sus propios objetos. Por ejemplo, un objeto ventana es creado (con ayuda del administrador de objetos) y gestionado por el subsistema Win32. La razón de no incluir la gestión de ese objeto en el Executive es que una ventana sólo es innata de las aplicaciones Windows, y no de las aplicaciones UNIX o OS/2. Por tanto, el Executive no se encarga de administrar los objetos relacionados con el entorno de cada S.O. concreto, sino de los objetos comunes a los tres.

El administrador de procesos (Process Manager). Se encarga (en colaboración con el administrador de objetos) de crear, destruir y gestionar los procesos y subprocessos. Una de sus funciones es la de repartir el tiempo de CPU entre los distintos subprocessos. Suministra sólo las relaciones más básicas entre procesos y subprocessos, dejando el resto de las interrelaciones entre ellos a cada subsistema protegido concreto. Por ejemplo, en el entorno POSIX existe una relación filial entre los procesos que no existe en Win32, de manera que se constituye una jerarquía de procesos. Como esto sólo es específico de ese subsistema, el administrador de objetos no se entromete en ese trabajo y lo deja en manos del subsistema.

El administrador de memoria virtual (Virtual Memory Manager). Windows NT y UNIX implementan un direccionamiento lineal de 32 bits y memoria virtual paginada bajo demanda. El VMM se encarga de todo lo relacionado con la política de gestión de la memoria. Determina los conjuntos de trabajo de cada proceso, mantiene un conjunto de páginas libres, elige páginas víctima, sube y baja páginas entre la memoria RAM y el archivo de intercambio en disco, etc.

El administrador de entrada salida (I/O Manager). Consta de varios subcomponentes: el administrador del sistema de ficheros, el servidor de red, el redirector de red, los drivers de dispositivo del sistema y el administrador de cachés. Buena parte de su trabajo es la gestión de la comunicación entre los distintos drivers de dispositivo, para lo cual implementa una interfaz bien definida que permite el tratamiento de todos los drivers de una manera homogénea, sin preocuparse del funcionamiento específico de cada uno. Trabaja en conjunción con otros componentes del Executive, sobre todo con el VMM. Le proporciona la

E/S síncrona y asíncrona, la E/S a archivos asignados en memoria y las caches de los ficheros. El administrador de caches no se limita a gestionar unos cuantos buffers de tamaño fijo para cada fichero abierto, sino que es capaz de estudiar las estadísticas sobre la carga del sistema y variar dinámicamente esos tamaños de acuerdo con la carga. El VMM realiza algo parecido en su trabajo.

El monitor de referencias a seguridad. Este componente da soporte en modo privilegiado al subsistema de seguridad, con el que interacciona. Su misión es actuar de alguna manera como supervisor de accesos, ya que comprueba si un proceso determinado tiene permisos para acceder a un objeto determinado, y monitoriza sus acciones sobre dicho objeto. De esta manera es capaz de generar los mensajes de auditorías. Soporta las validaciones de acceso que realiza el subsistema de seguridad local.

El núcleo (Kernel). Situado en el corazón de Windows NT, se trata de un micro-kernel que se encarga de las funciones más básicas de todo el sistema operativo: ejecución de subprocessos, sincronización multiprocesador, manejo de las interrupciones hardware.

El nivel de abstracción de hardware (HAL). Es una capa de software incluida en el Executive que sirve de interfaz entre los distintos drivers de dispositivo y el resto del sistema operativo. Con el HAL, los dispositivos se presentan al S.O. como un conjunto homogéneo con el cual interacciona a través de un conjunto de funciones bien definidas. Estas funciones son llamadas tanto desde el S.O. como desde los propios drivers. Permite a los drivers de dispositivo adaptarse a distintas arquitecturas de E/S sin tener que ser modificados en gran medida. Además oculta los detalles hardware que conlleva el multiprocesamiento simétrico de los niveles superiores del S.O.

Llamadas a procedimientos locales y remotos. Windows NT, al tener una arquitectura cliente-servidor, implementa el mecanismo de llamada a procedimiento remoto (RPC) como medio de comunicación entre procesos clientes y servidores, situados ambos en máquinas distintas de la misma red. Para clientes y servidores dentro de la misma máquina, la RPC toma la forma de llamada a procedimiento local (LPC).

Llamada a Procedimiento Remoto (Remote Procedure Call -RPC). Se puede decir que el sueño de los diseñadores de Windows NT es que algún día se convierta en un sistema distribuido puro, es decir, que cualquiera de sus componentes pueda residir en máquinas distintas, siendo el kernel en cada máquina el coordinador general de mensajes entre los distintos componentes. En la última versión de Windows NT esto no es aún posible. No obstante, el mecanismo de RPC permite a un proceso cliente acceder a una función situada en el espacio virtual de direcciones de otro proceso servidor situado en otra máquina de una manera totalmente transparente. Vamos a explicar el proceso en conjunto.

Llamada a procedimiento local (Local Procedure Call –LPC). Se usan cuando un proceso necesita los servicios de algún subsistema protegido, típicamente Win32. El subsistema Win32 va guardando en su propio espacio de direcciones una lista con todos los objetos que le van pidiendo los procesos. Por consiguiente, los procesos no tienen acceso a la memoria donde están los objetos; simplemente obtienen un descriptor para trabajar con ellos.

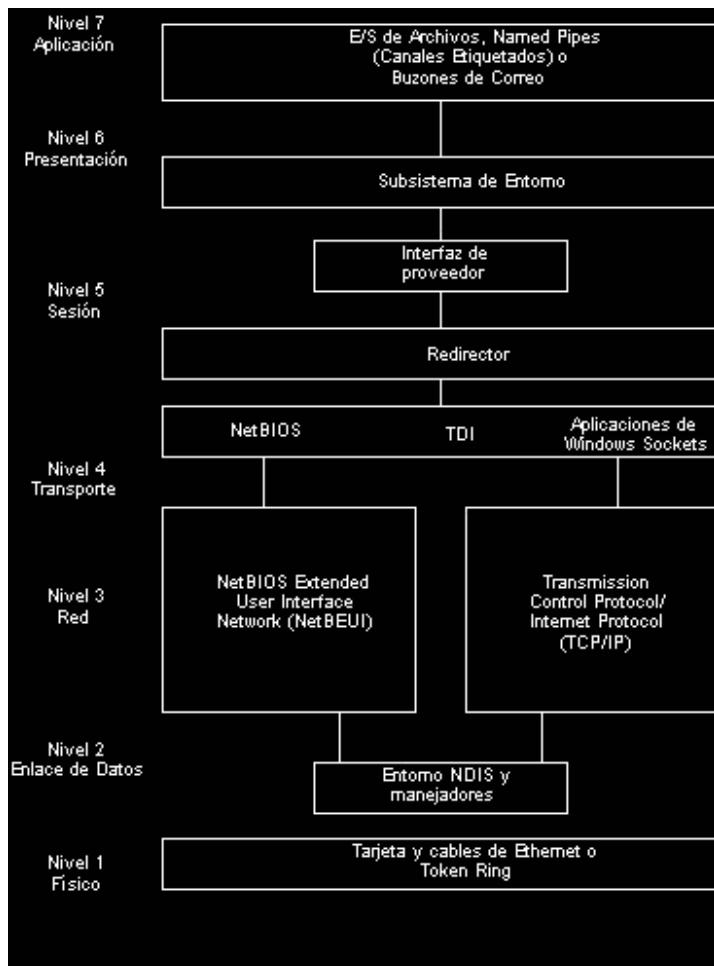
1.6.2 Windows NT Server

Windows NT Server es un sistema operativo para servidores, ampliable e independiente de la plataforma. Puede ejecutarse en sistemas basados en procesadores Intel x86, RISC y DEC Alpha, ofreciendo al usuario mayor libertad a la hora de elegir sus sistemas informáticos. Es ampliable a sistemas de multiproceso simétrico, lo que permite incorporar procesadores adicionales cuando se desee aumentar el rendimiento.

Internamente posee una arquitectura de 32 bits. Su modelo de memoria lineal de 32 bits elimina los segmentos de memoria de 64 KB y la barrera de 640 KB de MS-DOS.

Posee múltiples threads (subprocesos) de ejecución, lo que permite utilizar aplicaciones más potentes. La protección de la memoria garantiza la estabilidad mediante la asignación de áreas de memoria independientes para el sistema operativo y para las aplicaciones, con el fin de impedir la alteración de los datos. La capacidad de multitarea de asignación prioritaria permite al sistema operativo asignar tiempo de proceso a cada aplicación de forma eficaz.

Arquitectura de redes abiertas. Windows NT Server es compatible con los estándares NDIS (Especificación de la interfaz del controlador de red) y TDI (Interfaz del controlador de transporte). NDIS es una interfaz estándar para comunicación entre controladores de tarjetas adaptadoras de red y protocolos de red. NDIS le permite combinar y coordinar tarjetas y protocolos de red sin que sea necesario disponer de una versión diferente del protocolo de red para cada tipo de tarjeta. Permite también utilizar varios protocolos en una misma tarjeta de red. Con Windows NT Server se suministran cuatro protocolos compatibles con el estándar NDIS: TCP/IP, Microsoft NWLink, NetBEUI y DLC (Control de vínculos de datos). La interfaz TDI se comunica entre el protocolo de red y el software de red de alto nivel (como el servidor y el redirector). TDI elimina la necesidad de que el redirector y el servidor se comuniquen directamente con los protocolos de red, o de tener información de los mismos, permitiendo de esta forma utilizar protocolos, servidores o redirectores diferentes con Windows NT Server. También es compatible con aplicaciones de RPC (Llamada a procedimientos remotos), aplicaciones de sistema de entrada/salida básico de red (NetBIOS) y aplicaciones con Windows Sockets.

Gráfica 95. Arquitectura de Windows NT con Advanced Server

Seguridad incorporada. Windows NT Server incorpora la seguridad en el sistema operativo. El control de acceso discrecional le permite asignar permisos a archivos individuales. El concepto de derechos de usuario le ofrece un sistema de control discrecional de las funciones básicas del sistema, como establecer la hora y cerrar la computadora. Se incluyen, asimismo, funciones completas de auditoría.

Windows NT Server permite crear dominios y establecer relaciones de confianza, con el fin de centralizar las cuentas de usuario de la red y otro tipo de información de seguridad, facilitando el uso y la administración de la red. Con una administración centralizada de la seguridad, sólo es necesario administrar una cuenta por cada usuario. Dicha cuenta permite al usuario acceder a todos los recursos de la red.

Registro de configuración. Windows NT Server y Windows NT Workstation mantienen una base de datos denominada Registro. Esta base de datos contiene información acerca del sistema operativo, de la computadora y de los usuarios que anteriormente hayan iniciado sesiones en esta computadora. Las aplicaciones que

detecten la presencia de Windows NT podrán almacenar en el Registro la información de inicialización.

El Registro reemplaza la necesidad de separar los archivos de configuración. Sin embargo, para ser compatible con aplicaciones escritas para utilizar CONFIG.SYS y AUTOEXEC.BAT, Windows NT automáticamente mantiene y usa versiones de estos archivos que contienen solamente la información de la aplicación.

Administración de las estaciones de trabajo de los usuarios. Los perfiles de usuario de Windows NT Server le permiten proporcionar mayor facilidad de uso a los usuarios y al mismo tiempo restringir sus actividades en las estaciones de trabajo. Si desea utilizar perfiles para aumentar la productividad de los usuarios, puede guardar en los servidores un perfil con la configuración y las preferencias de los usuarios, tales como las conexiones de red, los grupos de programas e incluso los colores de la pantalla. Este perfil se utilizará cada vez que el usuario inicie una sesión en cualquier computadora con Windows NT, de forma que el entorno definido por el usuario le siga de una estación de trabajo a otra. Si desea utilizar los perfiles de usuario para limitar las actividades de los usuarios, deberá agregar restricciones al perfil, como por ejemplo, impedir que el usuario cambie los grupos y los elementos de programas que usted haya definido, o inhabilitar parte de la interfaz de Windows NT cuando el usuario haya iniciado una sesión.

Monitorización del rendimiento. Windows NT Server incluye también una sofisticada aplicación que permite monitorizar el rendimiento. Puede utilizar esta herramienta para observar, representar gráficamente y registrar cientos de datos estadísticos acerca de tipos específicos de rendimiento, agrupados en categorías generales tales como tráfico entre servidores de la red, rendimiento de los discos, uso de los procesadores, y estadísticas de los servidores y las estaciones de trabajo.

El Monitor de sistema le permite supervisar simultáneamente el rendimiento de un gran número de computadoras remotas, de forma que pueda controlar y comparar simultáneamente el rendimiento y el uso de un gran número de servidores.

Seguimiento de la actividad de la red. Windows NT Server proporciona numerosas herramientas para realizar el seguimiento de la actividad y el uso de la red. Puede observar los servidores y examinar qué recursos están compartiendo, ver qué usuarios están conectados a un servidor de la red y observar qué archivos tienen abiertos, registrar y ver las anotaciones de auditoría de seguridad, mantener registros de error exhaustivos y especificar las alertas que se deben enviar a los administradores en caso de que se produzcan determinados sucesos. Si su red utiliza el protocolo TCP/IP, podrá emplear también la utilidad de administración SNMP, suministrada con Windows NT Server.

1.6.3 Ventajas de Windows NT

- La instalación es muy sencilla y no requiere de mucha experiencia.
- Es multitarea y multiusuario.
- Apoya el uso de múltiples procesadores.
- Soporta diferentes arquitecturas.
- Permite el uso de servidores no dedicados.
- Soporta acceso remoto, ofreciendo la detección de intrusos, y mucha seguridad en estas sesiones remotas.
- Apoyo para archivos de DOS y MAC en el servidor.
- El sistema está protegido del acceso ilegal a las aplicaciones en las diferentes configuraciones.
- Permite cambiar periódicamente las contraseñas.
- Soporta múltiples protocolos.
- Carga automáticamente manejadores en las estaciones de trabajo.
- Trabaja con impresoras de estaciones remotas.
- Soporta múltiples impresoras y asigna prioridades a las colas de impresión.
- Muestra estadísticas de Errores del sistema, Caché, Información Del disco duro, Información de Manejadores, Nº de archivos abiertos, Porcentaje de uso del CPU, Información general del servidor y de las estaciones de trabajo, etc.
- Brinda la posibilidad de asignar diferentes permisos a los diferentes tipos de usuarios.
- Permite realizar diferentes tipos de auditorías, tales como del acceso a archivos, conexión y desconexión, encendido y apagado del sistema, errores del sistema, información de archivos y directorios, etc.
- No permite criptografía de llave pública ni privada.
- No permite realizar algunas tareas en sesiones remotas, como instalación y actualización.

1.6.4 Desventajas de Windows NT

- Tiene ciertas limitaciones por RAM, como: Nº Máximo de archivos abiertos y almacenamiento de disco total.
- Requiere como mínimo 64 Mb en RAM y un procesador Pentium de 133 MHz o uno superior.
- El usuario no puede limitar la cantidad de espacio en el disco duro.
- No soporta archivos de NFS.
- No ofrece el bloqueo de intrusos.
- No soporta la ejecución de algunas aplicaciones para DOS.

1.6.5 Características

- Es nueva tecnología para el mundo de las PC y es diferente por su ambiente gráfico, pero realmente no es nueva tecnología.
- Está basado en variaciones del kernel de Mac de UNIX. La arquitectura del microkernel soporta aplicaciones no diseñadas para Windows NT.
- Operaciones básicas de sistemas y otras capas sobre ella.
- Soporta 5 subsistemas: Windows 32 bits / Windows 16 bits / DOS / POSIX / OS/2.
- Funciona como Cliente – Servidor en un ambiente de red.
- Permite desarrollar servicios de redirecciónamiento para LAN Manager de Mips, RISC y Digital Alpha.
- Soporta sistemas de multiproceso.
- Cada aplicación se encuentra ejecutando en un hilo tratado como una caja multiprocesadora.
- Al igual que OS/2 ejecuta aplicaciones con errores de codificación, principalmente al ejecutarse en procesadores 386 y 486.
- Cada aplicación es limitada a un espacio de memoria (Esquema de direccionamiento de 32 bits real). Ejecuta aplicaciones de 16 y 32 bits y de otros Sistemas Operativos y para RISC de 64 bits.
- Existe una versión para Laptop.
- Soporta la tecnología Plug-in para sistemas API y sistemas de archivos instalables.
- También cuenta con servicios básicos de redes y APIs para archivos, manejadores de impresión, manejo de mensajes y seguridad directa. Aplicaciones para redes digitales que pueden ejecutarse en diferentes plataformas.
- Implantó facilidades para el uso de OSF, DCE y RPCs.
- Para facilitar los puertos de aplicación aísla el kernel del Hardware (Tipo de interfaz para el Sistema Operativo), con lo que se logra la portabilidad o compatibilidad a nivel de código.
- Provee datos, aplicaciones y protección del sistema contra accesos inadvertidos.
- Permite a los usuarios un acceso seguro a más información sin comprometer la seguridad del sistema.
- Conserva las principales características del servidor 3.51 incluso el protocolo nativo NetBEUI, IPX y TCP/IP.
- Soporta hasta 256 usuarios, administración de multidominio y replicación de directorio.
- Nuevas o mejoradas herramientas de administración y perfeccionamiento en la ejecución.
- El servidor NT relacionado con Internet, envía la información con el servidor de Internet IIS versión 2.0. También hace uso del FTP. Relaciona nuevos rasgos punto a punto con el protocolo PPTP y TCP/IP.

- Ayuda a consolidar la posición de NT como la plataforma del servidor en escenarios de Internet.
- Adopta el estilo de Unix de servicio de dominio DNS como norma.
- Incluye herramientas basadas en el Web referentes a la administración.

Seguridad. Windows NT ofrece gran seguridad por medio del acceso por cuentas y contraseñas. Es decir un usuario debe tener su cuenta asignada y una contraseña para poder tener acceso al sistema.

Contiene protecciones para directorios, archivos, y periféricos, es decir que todo esto se encuentra con una contraseña para poder ser utilizados.

Concepto de derechos. Permite a un grupo de usuarios efectuar determinadas operaciones.

Cuenta administrador. Controla todos los permisos y con ellas se puede: dar de alta; asignar cuentas; cancelar derechos.

Comunicación. Permite acceder y compartir discos en red. Permite compartir archivos, directorios y periféricos.

Sistemas de Archivos. Tiene 3 diferentes tipos y uno nuevo desarrollado por NT. Los primeros 3 son para compatibilidad: FAT para DOS; HPFS para OS/2; CDFS se usa para acceder discos compactos; NTFS es el sistema de archivos propio de Windows NT, el cual está basado en un sistema de transacciones, es decir que tiene la capacidad de almacenar una gran cantidad de operaciones a disco para que en el caso de alguna falla este elemento pueda ser usado para la reconstrucción del sistema de archivos del disco.

Multitarea. Para la ejecución simultánea de múltiples tareas NT utiliza: Manager; Scheduler; Manejador de excepciones e interrupciones. Mecanismos de sincronización. El usuario puede dejar ejecutando alguna tarea en una ventana y seguir trabajando en otra.

Memoria Virtual. NT tiene un manejador de memoria virtual que permite el uso de un espacio de direccionamiento de 2 GB. Este espacio de direccionamiento esta protegido de otros procesos del sistema. Traduce direcciones virtuales a direcciones físicas. Y también se encarga del problema de traer y llevar páginas de disco a memoria y de memoria a disco.

Protocolos que soporta. Son los siguientes:

- NetBEUI.
- TCP/IP.
- IPX/SPX.

- Banyan
- DECnet.
- Apple Talk.

Funcionamiento de TCP/IP. Protocolo de control de transmisión/Protocolo Internet. Fue desarrollado a finales de los años 70, como resultado de un proyecto de investigación sobre interconexión de redes realizado por la Agencia de proyectos de investigación avanzada para la defensa (DARPA) de Estados Unidos. La principal ventaja y utilidad de TCP/IP es que es un protocolo estándar y reencaminable; se trata del protocolo más completo y aceptado de todos los existentes. Permite comunicarse a través de redes interconectadas con distintos sistemas operativos y arquitecturas de hardware, como UNIX o computadoras principales, así como con Windows NT.

Permite comunicarse a través de redes interconectadas con distintos sistemas operativos y arquitecturas de hardware, como UNIX o computadoras principales, así como con Windows NT.

TCP/IP de Microsoft utiliza también la interfaz de NetBIOS, comúnmente conocida como petición para comentarios (RFC) de NetBIOS. Además, Microsoft proporciona diversas utilidades TCP/IP para su uso con TCP/IP en Windows NT. Es el protocolo más aceptado, aunque no es tan rápido como NetBEUI en redes locales de pequeño tamaño.

Funcionamiento de NetBEUI. NetBEUI (Interfaz extendida de usuario de NetBIOS) fue presentado por primera vez por IBM en 1985. NetBEUI es un protocolo compacto, eficiente y rápido.

NetBEUI está optimizado para obtener un rendimiento muy elevado cuando se utiliza en redes locales o segmentos de redes locales departamentales. En cuanto al tráfico cursado dentro de un segmento de red local, NetBEUI es el más rápido de los protocolos suministrados con Windows NT.

Funcionamiento de DLC (Control de vínculo de datos). A diferencia de NetBEUI y TCP/IP, el protocolo DLC no ha sido diseñado para servir de protocolo principal entre PC. Por el contrario, se suele utilizar DLC con Windows NT si se necesita que las computadoras con Windows NT accedan a computadoras principales IBM o si se está configurando una impresora que se conecta directamente a un cable de red.

Si se desea utilizar DLC para permitir la comunicación entre computadoras con Windows NT y computadoras principales, bastará con añadir el protocolo DLC como protocolo adicional en cada una de las computadoras que se comunican realmente con las computadoras principales. No será necesario que instale DLC en todas las computadoras de la red.

1.7 Instalación Windows XP/2003

Para instalar Windows XP Professional se necesita, como mínimo, la siguiente configuración de memoria RAM de 64 MB, un disco duro de 2 GB con controladora IDE, una tarjeta Ethernet de 10/100 Mbps y CD – ROM.

Sin embargo, la recomendación que se realiza para la configuración de las estaciones de trabajo es de ordenadores con un procesador Pentium II a 350 Mhz o superior, una configuración de memoria RAM de 128 MB y un disco duro de entre 8 y 20 GB.

Para Windows 2003 se deben seguir las recomendaciones del punto 1.4.4 Requisitos del sistema Windows 2003.

Aunque ambos (XP/2003) chequean automáticamente el hardware antes de su instalación para informar de cualquier conflicto que pudiera existir, puede asegurarse previamente de que no va a encontrar ningún conflicto si consulta la **lista de compatibilidad de hardware (HCL)**, que se encuentra en la dirección <http://www.microsoft.com/windows/catalog/server/> (con las últimas actualizaciones hardware y software). También se puede comprobar desde la opción comprobar compatibilidad del sistema del propio CD de la instalación.

Incorpora la tecnología Plug and Play que permite que el sistema operativo reconozca automáticamente cualquier dispositivo que incorpore dicha tecnología (de esta manera, se resolverán los problemas que pudieran existir referidos a su configuración).

Si no sabe si cuenta con dispositivos que no incorporen esta tecnología en su equipo, deberá realizar un inventario previo para averiguar si dispone de dichos dispositivos y, en caso de que sea así, anotar la dirección de memoria y la interrupción (IRQ) que están utilizando todos los dispositivos. Una vez anotado deberá comprobar que no hay conflictos entre las direcciones de memoria y las play como los que no lo hacen y, en caso de haber conflictos, deberá corregir manualmente los conflictos existentes.

1.7.1 Consideraciones previas de la instalación

Antes de realizar una instalación nueva es necesario tener en cuenta las siguientes consideraciones:

- El modo de licencia que se va a utilizar. Windows Server 2003 soporta dos tipos de licencia.
- **Por servidor.** En este tipo, se asigna un número de licencias al servidor que permitirá realizar conexiones con un determinado producto de

servidor. Cuando un usuario se conecte con el producto de servidor en dicho servidor, la conexión consume una licencia que, cuando dicho usuario se desconecte, quedará libre para su utilización por otro usuario. Es necesario, como mínimo, disponer del mismo número de licencias en cada servidor que de estaciones vayan a conectarse en un momento dado. Suele ser la instalación más económica para redes en que las estaciones se conectan a un único servidor.

- **Por dispositivo o por usuario.** En este tipo se requiere una licencia de acceso de cliente por cada dispositivo o usuario que se conecte al producto del servidor seleccionado. Una vez que el dispositivo disponga de una licencia puede tener acceso a cualquier servidor en el que se ejecute dicho producto. Suele ser la más económica si se realizan conexiones a más de un servidor. Se necesita una organización de acceso de cliente para cada equipo que tenga acceso al servidor para servicios básicos de red.

En caso de no estar seguro del modo de licencia que se necesita es preferible seleccionar **por servidor** ya que se puede cambiar posteriormente sin ningún problema.

- Si se desea una configuración de arranque dual, es decir, si se va a poder escoger entre dos o más sistemas operativos cuando se inicie el ordenador (la principal desventaja de una configuración de arranque doble es poder utilizar aplicaciones que únicamente se ejecuten en un sistema operativo). Es necesario tener en cuenta que no es posible trabajar con discos dinámicos con Windows Server 2003 si se tiene una configuración de arranque dual.

Entre las posibles configuraciones de arranque dual se encuentran las siguientes:

- En un arranque dual con **Windows 95** y Windows XP/2003 se deben cumplir los siguientes requisitos:
 - La participación primaria debe estar formateada como FAT o FAT32 (si es Windows 95 OSR2).
 - WindowsXP/2003 se debe instalar después de Windows 95 y en una partición distinta.
 - Los volúmenes comprimidos con DriveSpace o DoubleSpace no estarán disponibles cuando se ejecute Windows XP/2003.

- En un arranque dual con **Windows 98** y Windows XP/2003 se debe cumplir los siguientes requisitos:
 - La partición primaria debe estar formateada como FAT o FAT32.
 - Windows XP/2003 se debe instalar después de Windows 95 y en una partición distinta.
 - Los volúmenes comprimidos con DriveSpace o DoubleSpace no estarán disponibles cuando se ejecute Windows XP/2003.
- En un arranque dual con **Windows NT 4 o Windows 2000** y Windows XP/2003 se deben cumplir los siguientes requisitos:
 - No es recomendable instalar ambos sistemas operativos con NTFS.
 - Asegúrese de tener instalado el último Service Pack de Windows NT o Windows 2000.
 - Instale cada uno de los sistemas operativos en un disco distinto o en particiones distintas.
 - No instale Windows XP/2003 en una unidad comprimida (a no ser que haya sido comprimida con NTFS)
 - Instale los programas que va a utilizar con cada sistema operativo en su partición correspondiente (si desea instalar los mismos programas para ambos, deberá instalarlo dos veces).
 - Si ambos van a estar en el mismo dominio, cada instalación deberá hacerse con un nombre distinto cada vez.
- Qué sistema de archivos se va a utilizar. Es posible escoger entre tres sistemas de archivos distintos para las particiones del disco.
 - **FAT (File Allocation System).** Se puede acceder a este sistema de archivos desde MS-DOS y todas las versiones de Windows. Permite trabajar con particiones menores de 2 GB y no soporta dominios.
 - **FAT32.** Se puede acceder a este sistema de archivos desde Windows 95 OSR2, Windows 98, Windows 2000, Windows XP y Windows Server 2003. Permite trabajar con particiones mayores de 2 GB, el tamaño máximo de un archivo es de 4 GB, los volúmenes pueden llegar hasta 2 TB (en Windows 2000 solo hasta 32 GB) y no soporta dominios.
 - **NTFS (NT File System).** Es el sistema desarrollado para Windows NT 4 que permite nombres de archivos de hasta doscientos cincuenta y seis caracteres, ordenación de directorios, atributos de acceso a archivos, reparto de unidades en varios discos duros, reflexión de discos duros y registro de actividades. En Windows 2000 Server se incluyeron mejoras

que permiten utilizar el Directorio Activo, dominios, cuotas de discos para cada usuario, cifrado y compresión de archivos, almacenamiento remoto, una herramienta de desfragmentación y utilización de enlaces de archivos similares a los realizados en UNIX. Sus volúmenes pueden llegar a 16 TB menos 64 KB y el tamaño máximo de un archivo sólo está limitado por el tamaño del volumen.

Cuando se está trabajando con nombres de archivo de formato largo (NTFS) en el servidor y se tienen que convertir automáticamente a un alias en formato **MS-DOS**, se utiliza el siguiente método:

- Se eliminan espacios.
- Se mantienen los seis primeros caracteres del nombre.
- Se añade el carácter ~ seguido del número del uno al cuatro.
- Se mantiene la extensión existente.

Por ejemplo, los nombres largos se convertirán de la forma siguiente en cortos:

Informe anual año 1991.doc	INFORM~1.DOC
Informe anual año 1992.doc	INFORM~2.DOC
Informe anual año 1993.doc	INFORM~3.DOC
Informe anual año 1994.doc	INFORM~4.DOC

Pero cuando se halla llegado a este nivel, ya no se puede mantener el mismo método y se modifica al siguiente:

- Se eliminan espacios
- Se mantienen los dos primeros caracteres del nombre
- Se generan los cuatro caracteres siguientes desde el tres hasta el seis
- Se añade al carácter ~ seguido del número uno (a no ser que en el proceso anterior hubiera resultado unos caracteres que ya estuvieran en otro archivo anterior en cuyo caso pondría el número dos).
- Se mantiene la extensión existente

Por ejemplo, los nombres largos se seguirán convirtiendo de la forma siguiente en cortos:

Informe anual año 1995.doc	INA4F5~1.DOC
Informe anual año 1996.doc	INE5A7~1.DOC
Informe anual año 1997.doc	INE5A7~2.DOC

En qué partición se instalará o si se necesita crear una nueva partición. Es necesario tener planificadas las particiones del disco que se van a utilizar antes de

proceder a la instalación de Windows XP/2003, en dicha planificación es necesario tener en cuenta:

- Las particiones son la manera en que se divide el disco físico, de forma que cada una de ellas funcionan como si fueran unidades separadas.
- Cada partición se puede formatear en un sistema e archivo distinto y se pueden designar con una letra de una unidad distinta. (C: D: etc.)
- Cuando se formatea una partición, todos los archivos que hubiera previamente se destruyen.
- La partición primaria (o del sistema) es aquella en donde se instalan los archivos que van a cargar el sistema operativo.
- Es preciso determinar el tamaño de la partición donde se va a instalar Windows XP/2003 antes de realizar la instalación.
- No instale Windows XP/2003 en una unidad comprimida a no ser que haya sido comprimida con el sistema de compresión de NTFS.
- Durante el proceso de la instalación se creará la partición donde se instalará el sistema operativo. Una vez finalizada esta, se podrán administrar los discos utilizando las herramientas que lleva incorporadas.
- No es posible trabajar con discos dinámicos en una configuración de arranque dual.
- Se van a utilizar los servicios de *instalación remota*, se necesitará una partición distinta para ellos.
- Es conveniente colocar únicamente el sistema operativo en una partición (o en un disco) y los archivos de datos y aplicaciones por separado.

Qué componentes se van a instalar. Windows Server 2003 incorpora una serie de componentes que se añaden automáticamente durante la instalación. Además, se pueden escoger otros componentes que aumentan sus posibilidades (en caso de no hacerlo durante la instalación, se podrá hacer posteriormente desde agregar o quitar componentes de Windows de agregar o quitar programas del panel de control). O desde instalar componentes adicionales desde el CD de instalación.

Estos componentes son los siguientes:

- **Accesorios y utilidades.** Incluye accesorios y utilidades de Windows como WordPad, Saint, calculadora, comunicaciones, etc.

- **Actualización de certificados raíz.** Descarga automáticamente los certificados raíz más actuales para obtener seguridad de correo electrónico, exploración y distribución de software.
- **Configuración de seguridad mejorada de Internet Explorer.** Limita la forma en que los usuarios exploran los sitios Web de Internet.
- **Herramientas de administración y supervisión.** Incluye herramientas para supervisar y mejorar el rendimiento de la red (por ejemplo SNMP).
- **Licencias de Terminal Server.** Permite configurar el equipo como servidor de licencias de Terminal Server y proporcionar licencias de cliente.
- **Otros servicios de impresión y archivo en red.** Permite compartir archivos o impresoras en este equipo con otros de la red que utilicen Macintosh o UNIX.
- **Servicios de Index Server.** Proporciona funciones de indexación para los documentos almacenados en disco permitiendo, a los usuarios, la búsqueda rápida de texto o propiedades.
- **Servicios de Certificate Server.** Instala una entidad emisora para emitir certificados que se utilicen con programas de seguridad de claves públicas.
- **Servicio de correo electrónico.** Permite instalar el servicio e administración de POP3 para recuperación del correo electrónico recibido.
- **Servicios de fax.** Recibir y enviar.
- **Servicio de instalación remota.** Ofrece la posibilidad de instalar estaciones de trabajo de forma remota.

Servicios de red. Proporcionan soporte para las comunicaciones de red incluyendo:

- **Protocolo de configuración dinámica de Host (DHCP).** Permite asignar direcciones IP dinámicas a distintos dispositivos de red. (servidores, estaciones, impresoras, escáneres)
- **RPC sobre el Proxy HTTP.** Habilita. RPC/DCOM para desplazarse por HTTP a través de IIS.

- **Servicio de autenticación de Internet.** Proporciona autenticación, autorización y contabilidad de usuarios de acceso telefónico y VPN (incluye soporte protocolo RADIUS).
- **Servicio WINS.** Proporciona resolución de nombres para clientes que ejecuta Windows NT o versiones de anteriores de otros sistemas operativos Windows.
- **Servicios simples de TCP/IP.** Admite los siguientes servicios de TCP/IP: generador de caracteres, hora diurna, desechar, eco y cita del día.
- **Sistema de nombres de dominio (DNS).** Proporciona resolución de nombres para clientes que ejecutan Windows 2000, Windows XP o Windows Server 2000.

Servicios de Windows Media. Proporcionan soporte multimedia que permiten enviar contenidos digitales a través de la red

- **Servicios UDDI.** Instala los servicios (Universal Description Discovery and Integration) que permiten publicar y buscar información acerca de los servicios Web.
- **Servidor de aplicaciones.** Permite instalar ASP.NET, IIS, Mesage, Queue Server y la consola del servidor de aplicaciones.
- **Terminal Server.** Configura el equipo para permitir que varios usuarios ejecuten una o más aplicaciones de forma remota.

Qué configuración de direcciones se va a utilizar

Como se explicó en Windows NT, TCP/IP es un protocolo que proporciona acceso a Internet, y por ello, necesita una dirección IP por cada ordenador (se ha de indicar en las propiedades del protocolo TCP/IP). Para proporcionársela se pueden seguir los siguientes métodos de configuración:

- **Configuración manual o estática.** Se utilizará este método de configuración cuando se disponga de una red con múltiples segmentos y no se cuente con un servidor DHCP. Será necesario indicar una dirección IP, una máscara de subred, la puerta de enlace predeterminada, el servidor DNS y/o el servidor WINS en cada uno de los equipos (incluido el servidor).
- **Configuración automática o dinámica.** Con este método se asignará automáticamente una dirección IP al equipo. Hay dos formas posibles:

- **Configuración automática.** Se utilizará este tipo de configuración cuando se disponga de una red pequeña con pocos servidores sin necesidad de conexión a Internet y no se cuente con un servidor DHCP. Se deberá indicar que se desea obtener una asignación automática de dirección IP, y al no encontrar un servidor DHCP, Windows asignará la dirección IP utilizando **APIPA (Automatic Private IP Addressing)**. Esta asignación se realizará en el rango de direcciones 169.254.0.1-169.254.255.254 y con la máscara de subred 255.255.0.0 (no es necesario indicar la puerta de enlace predeterminada, el servidor DNS o el servidor WINS).
- **configuración dinámica.** Este tipo de configuración se utilizará en una red que disponga de un servidor DCHP. Se deberá indicar que se desea obtener una asignación automática de dirección IP y, al encontrar DCHP, éste asignará una dirección IP, una máscara de subred, la puerta de enlace predeterminada, el servidor DNS y/o el servidor WINS a cada uno de los equipos cuando se conecten.

Qué resolución de nombres desea

Como se ha visto en un apartado anterior, la resolución de nombres es un proceso que permite a los usuarios conectarse a la red utilizando el nombre de los equipos en lugar de su dirección IP (de esta manera, no es necesario tener que usar la dirección IP, que es más difícil de recordar). Windows 2000 Server proporciona dos métodos de resolución de nombres que pueden coexistir conjuntamente:

- **DNS.** Es un método que utiliza servidores distribuidos a lo largo de una red para resolver el nombre de un ordenador en su dirección IP. Se necesita DNS para correo electrónico de Internet, navegación por páginas Web, trabajar con el Directorio Activo y para los clientes que ejecutan Windows 2000 o Windows XP. Este método de resolución de nombres se instala automáticamente cuando se instala automáticamente cuando se crea un controlador de dominio (o se promociona un servidor a controlador de dominio) a no ser que se detecte que ya existe un servidor DNS.
- **WINS.** Si va disponer de clientes que ejecuten Windows NT u otros sistemas operativos de Windows distintos de Windows 2000 o Windows XP, será necesario utilizar este método de resolución de nombres. Es un componente opcional y deberá seleccionarse, si se considera necesario, durante el proceso de instalación (también es posible hacerlo posteriormente).

Si se van a crear dominios o grupos de trabajo para los servidores.

Un **dominio** es un conjunto de cuentas de usuario y recursos de red bajo un nombre sencillo que tiene establecidas fronteras de seguridad. Se necesita para instalar el directorio Activo en Windows Server 2003 y tener un servidor de red.

Un **grupo de trabajo** es una agrupación básica que, únicamente, se establece para ayudar a determinados usuarios a utilizar objetos compartidos (impresoras y carpetas). Se necesita para tener una red entre iguales con Windows XP Profesional.

En un grupo de trabajo los usuarios deben trabajar con varias contraseñas (una para cada recurso compartido) mientras que, en un dominio, los usuarios únicamente han de utilizar una contraseña para conectarse a la red y trabajar con los recursos que haya disponibles. Además, en un dominio se puede restringir a los usuarios la utilización de determinados recursos utilizando los permisos.

Los servidores dentro de un dominio pueden tomar uno de los siguientes papeles:

- **Controladores de dominio.** Pertenecen al dominio y contiene una copia de las cuentas de usuario y de otros datos del Directorio Activo. Es obligatorio que haya, al menos, un controlador de dominio.
- **Servidores miembro.** Pertenecen al dominio y no contienen una copia de las cuentas de usuario y de otros datos del directorio Activo. Se utilizan para almacenar los archivos y otros recursos de red.

Además de estos dos tipos, sin formar parte de ningún dominio puede haber **Servidores Independientes**. Pertenecerán a grupos de trabajo y se utilizarán para almacenar archivos y otros recursos de red.

Se recomienda utilizar siempre dominios en lugar de grupos de trabajo (exceptuando para redes muy pequeñas y con pocos usuarios).

1.7.2 Preparando la instalación

Antes de proceder con la instalación, es necesario realizar los pasos siguientes que van a preparar el ordenador para una instalación nueva:

- Hacer una copia de seguridad de los archivos que hubiera en el ordenador (si hay alguno o se consideran necesarios). Dicha copia se puede realizar en otro disco duro, una unidad de cinta u otro ordenador de la red.
- Descomprimir el disco donde se va a realizar la instalación (a no ser que se haya comprimido con **NTFS**).
- Desactivar el reflejado de discos (si lo hubiera).

- Desconectar el ordenador de cualquier dispositivo SAI (Sistema de alimentación ininterrumpida). Al desconectar el cable serie que conecta el SAI con el ordenador, se evitará que haya problemas en el proceso de detección automática de dispositivos que se realiza durante la instalación.

1.7.3 Procediendo con la instalación Windows Server 2003

Una instalación nueva se puede realizar en un equipo que dispone previamente de otro sistema operativo (para realizar una configuración de arranque dual) o en un equipo totalmente limpio. En ambos casos, el proceso de la instalación se puede comenzar de varias formas:

- Desde una unidad CD-ROM del propio equipo. Es la forma más común de iniciar la instalación y puede hacerse de dos maneras distintas:
 - Iniciando el ordenador desde la unidad CD-ROM de un ordenador limpio. Comenzará automáticamente la instalación, aunque previamente habrá de haberse colocado el CD de Windows Server 2003 en la unidad correspondiente y se habrá indicado en el SETUP o BIOS del ordenador que puede iniciarse desde una unidad CD-ROM.
 - Iniciando el ordenador desde otro sistema operativo. Una vez cargado dicho sistema operativo y colocado el CD de Windows Server 2003 en la unidad CD-ROM correspondiente, se pueden dar tres opciones:
 - Ejecutarse automáticamente el proceso de instalación.
 - Desde el explorador de Windows, ver el contenido del CD y comenzar la instalación.
 - Cambiarse a la letra correspondiente a la unidad CD-ROM, ver su contenido y comenzar la instalación.
- Desde un recurso compartido de la red y utilizando un equipo que ya cuenta con otro sistema operativo. Para ello, se puede actuar de dos maneras distintas:
 - Colocando el CD de Windows Server 2003 en una unidad CD-ROM compartida de un equipo de la red. Habrá que buscar dicho recurso compartido, ver su contenido y comenzar la instalación.
 - Copiando el contenido del CD de Windows Server 2003 en una carpeta compartida de un equipo de la red. Habrá que buscar dicho recurso compartido, ver su contenido y comenzar la instalación.

Con cualquier método de iniciar la instalación, si ésta no comienza automáticamente, se deberá ejecutar el archivo SETUP.EXE del directorio raíz del

CD de la instalación o el archivo WINNT32.EXE del subdirectorio I386 del CD de la instalación (desde Windows) o el archivo WINNT.EXE del subdirectorio I386 del CD de la instalación (desde MS-DOS).

Una vez ejecutado el proceso de instalación se deben seguir uno a uno los pasos indicados en cada pantalla hasta terminar. El asistente para la instalación es muy completo y no deja que el usuario se pierda en ninguno de los pasos señalados.

1.8 Resumen de los principales sistemas operativos Windows

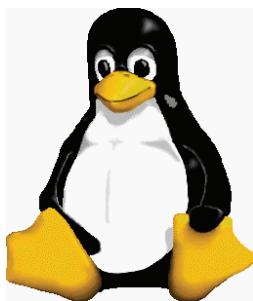
Sistema operativo	Descripción
Microsoft Windows 95	Este sistema operativo era un entorno multitarea dotado de una interfaz gráfica de usuario que, a diferencia de las versiones anteriores de Windows, no necesitaba que se cargara primero MS – DOS para ser ejecutado. Estaba basado en menús desplegables, ventanas y un dispositivo señalador (ratón). Una de las características principales de Windows 95 era que los nombres de los archivos no estaban restringidos a ocho caracteres y tres de la extensión, ya que podían tener hasta 256 caracteres para realizar una descripción completa del contenido del archivo. Además, incorporaba PLUG AND PLAY que es una tecnología desarrollada por los fabricantes de PCs, por la que un usuario puede instalar fácilmente los dispositivos de hardware, permitiendo al sistema realizarlo automáticamente sin la intervención de usuario.
Microsoft Windows 98	Es el siguiente escalón en la familia de sistemas operativos Windows. Como era obvio, esta nueva versión continúa soportando 32 bits y PLUG AND PLAY. Desde el punto de vista del usuario, Windows 98 no trae nada nuevo en su interfaz gráfica. Podría decirse que es una compilación de características, ya que muchas de ellas ya se encontraban en Internet Explorer 4.0 (al ser instalada esta actualización con Windows 95 OSR- 2). Permite soporte para FAT32 (al igual que Windows 95 OSR -2) y el Active Desktop (de IE 4). Podía instalarse una opción para poder trabajar como sistema multiusuario de estación de trabajo.
Microsoft Windows NT	Es un sistema operativo que brinda poder, velocidad y nuevas características; además de las tradicionales de los sistemas Windows. Fue lanzado al mercado el 24 de mayo de 1993) y es un sistema operativo de 32 bits que puede trabajar en procesadores 386, 486 y Pentium. Además de ser multitarea y multiusuario (en opción de servidor de redes), es multiprocesador (es decir, permite trabajar en un equipo que disponga de varios procesadores en la CPU). Se encuentra disponible en dos versiones: Estación de trabajo (versión Workstation) o Servidor (versión Server).
Windows Millennium	Soporta y comparte el mismo código que Windows 98 pero con una característica propia: dice adiós a MS – DOS, ya que no están disponibles las opciones de arranque solo Símbolo del Sistema y Reiniciar en modo MS–DOS. Los archivos de configuración conf.sys y autoexec.bat no se ejecutan (existen o no); solo tienen sentido durante la instalación de Windows; después dará igual su contenido. Por ello, no funcionará ningún programa MS–DOS (que necesite insertar parámetros en el archivo config.sys) y no es compatible 100% con aplicaciones diseñadas para MS–DOS.
Windows 2000	Es más rápido que Windows 98, ya que, a igualdad de memoria RAM, se ejecuta más rápido que Windows 98 y no se ralentiza con cargas pesadas. Los usuarios pueden ejecutar más programas y hacer más tareas al mismo tiempo porque Windows 2000 está basado totalmente en una arquitectura de 32 bits. Mejora el interfaz gráfico de Windows al reducir los iconos innecesarios del escritorio, simplifica el menú de inicio (introduciendo una nueva funcionalidad que adapta el menú de inicio a la manera de trabajo del usuario, mostrando sólo las aplicaciones que se utilizan más frecuentemente). Es más seguro que Windows NT, ya que está basado en un sistema de seguridad que permite a los usuarios o/y administradores seleccionar el nivel de protección apropiado para su información y aplicaciones, para intercambiar o almacenar información y aplicaciones, para intercambiar o almacenar información en ordenadores independientes, en la red, en una intranet o en Internet. Se encuentran disponibles cuatro versiones:
	➤ Professional. Es el sucesor de Windows NT Workstation. Permite utilizar hasta 2

Sistema operativo	Descripción
	<p>procesadores simétricos, hasta 4 GB de memoria RAM, herramientas de gestión de Windows, infraestructura de seguridad Kerberos y PKI, servicios de componentes y servicios de Internet.</p> <ul style="list-style-type: none"> ➤ Server. Es el sucesor de Windows NT Server. Permite utilizar hasta 4 procesadores, hasta 4 GB de memoria RAM e incorpora Directorio Activo, herramientas de gestión de Windows, infraestructura de seguridad Kerberos y PKI, servicios de terminales, servicios de componentes y servicios de Internet, balanceo de la carga de la red y servicios de cluster. ➤ Advanced Server. Permite utilizar 8 procesadores, hasta 8 Gb de memoria RAM e incorpora Directorio Activo, herramientas de gestión de Windows, insfraestructura de seguridad Kerberos y PKI, servicios de terminales, servicios de componentes, servicios de Internet, balanceo de la carga de la red y servicios de cluster. ➤ Datacenter Server. Permite utilizar hasta 32 procesadores, hasta 64 GB de memoria RAM e incorpora Directorio Activo, herramientas de gestión de Windows, infraestructura de seguridad Kerberos y PKI, servicios de terminales, servicios de componentes, servicios de Internet, balanceo de la carga de la red y servicios de cluster avanzados.
Windows XP Profesional	Básicamente es Windows 2000 Professional, pero con un nuevo y mejorado aspecto, así como una mejora notable de las características multimedia e Internet. Incorpora Direct X 8 (para juegos 3D y efectos gráficos), Internet Explorer 6, Outlook Express 6 y Windows Media Player 8.

CAPÍTULO 2. SISTEMA OPERATIVO UNIX/LINUX

Actividad a desarrollar:

Al igual que con el SO Windows es importante que Usted instale el SO LINUX, revise cada una de las características aquí descritas y verifique su funcionamiento. La versión la definen en cada CEAD dependiendo del software disponible o deseado. Realizar un manual de instalación de la distribución de linux seleccionada. En grupos de trabajo. Probarla en el laboratorio.



En las unidades precedentes se han visto conceptos, principios y características de los sistemas operativos actuales. En este capítulo se tratará más en profundidad uno de los más extendidos e interesantes, el UNIX en sus diferentes versiones, pero haciendo hincapié en Linux, una de sus variantes más extendidas en la actualidad.

El estudio de LINUX es especialmente interesante ya que se puede ejecutar en más tipos de arquitecturas que cualquier otro S.O., de hecho actualmente se puede decir que hay una versión de LINUX para cada máquina del mercado.

2.1 Historia

En 1965, los laboratorios de telefonía Bell (una división de A T&T) trabajaban (junto con General Electric y el M.I.T. (Instituto Tecnológico de Massachusetts) en el desarrollo de Multics, un proyecto para realizar un sistema operativo de grandes prestaciones, multitarea y de tiempo compartido. Por diversas razones, Bell se separó del grupo y Ken Thompson y Dennis Ritchie (empleados de Bell Technologies) decidieron continuar independientemente con el proyecto para su empresa. Así pues, se dedicaron a desarrollar un sistema operativo que cumpliera con la mayor parte de los requisitos del proyecto anterior pero con una concepción diferente, mucho más simple.

En 1970, Ken Thompson implementó una primera versión escrita en ensamblador, en una PDP-7, y, a modo de burla hacia el Multics de la General Electric, lo llamó UNIX. Posteriormente, en 1973 Dennis Ritchie desarrolló junto a B.W. Kernighan el lenguaje de programación C y junto a Thompson rescribieron todo el código

UNIX con este lenguaje, lo que, como se verá más adelante lo impulsó a los niveles de popularidad actuales.

En 1977, UNIX se portó a otra arquitectura diferente a la de la PDP, gracias a que para recompilarlo sólo era necesario realizar los cambios pertinentes para adaptarlo a la nueva arquitectura, manteniendo intacto el resto del código. Este es el concepto de sistema abierto que lo hizo triunfar. Además, UNIX fue diseñado modularmente, es decir, se programaron multitud de sencillos módulos genéricos que una vez interconectados eran capaces de realizar tareas complejas. Esto permitió una fácil depuración de sus errores y facilitó la colaboración entre distintos equipos de desarrollo.

En resumen, UNIX es uno de los sistemas operativos más populares del mundo debido a su extenso soporte, distribución y, sobre todo, a su característica de sistema abierto. Originalmente fue desarrollado como sistema multitarea con tiempo compartido para mini ordenadores y mainframes y, desde entonces, se ha convertido en uno de los sistemas más utilizados a pesar de su, ocasionalmente, confusa interfaz con el usuario y el problema de su estandarización.

Por su parte, Linux es una versión de UNIX de libre distribución, inicialmente desarrollada por Linus Torvalds en la Universidad de Helsinki, en Finlandia. Ha sido desarrollado con la colaboración de muchos programadores y expertos de UNIX a lo largo y ancho del mundo, gracias a la existencia de Internet. Cualquier habitante del planeta puede acceder a Linux y desarrollar nuevos módulos o cambiarlo a su antojo.

El núcleo de Linux no utiliza ni una sola línea del código de A T &T o de cualquier otra fuente de propiedad comercial y buena parte del software para Linux se desarrolla bajo las reglas del proyecto de GNU de la Free Software Foundation. Cambridge, Massachusetts.

Inicialmente, Linux fue un proyecto de aficionado de Linus Torvalds. Se inspiraba en Minix, un pequeño UNIX de carácter fundamentalmente académico diseñado para microprocesadores de Intel de las familias 8086, 8088 y 80286 desarrollado por Andrew Tanenbaum. La intención de Linus Torvalds era la de mejorar MINIX para aprovechar al máximo las características avanzadas del procesador 80386 de Intel como son la conmutación de tareas en modo protegido y el coprocesador matemático.

«Comencé a utilizar el C tras escribir algunos drivers, y ciertamente se aceleró el desarrollo. En este punto, sentí que mi idea de hacer un "un Minix mejor" que Mini" se hacía más seria. Esperaba que algún día pudiese recompilar el gcc bajo Linux...»

... Dos meses de trabajo, hasta que tuve un driver de discos (con numerosos bugs, pero que parecía funcionar en mi PC) y un pequeño sistema de ficheros. Aquí tenía

ya la versión 0.01 [Finales de Agosto e de 1991]: no era muy agradable de usar sin el drive de disketes, y no hacia gran cosa. No pensé que alguien compilaría esa versión”.

El 5 de octubre de 1991, Linus anunció la primera versión «oficial» de Linux, La 0.02. Ya podía ejecutar bash (el shell de GNU) y gcc (el compilador de C de GNU), pero no hacía mucho más. La intención era ser un juguete para hackers. No había nada sobre soporte a usuarios, distribuciones, documentación ni nada parecido. Hoy, la comunidad de Linux aún trata estos asuntos de forma secundaria. Lo primero sigue siendo el desarrollo del kernel.

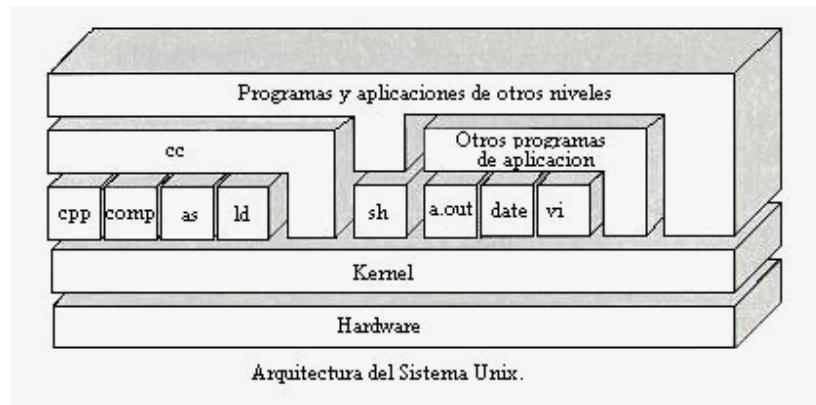
Actualmente, Linux es ya un clónico de UNIX completo, capaz de ejecutar X-Window, TCP/IP, Emacs, UUCP y software de correo y News, mucho software de libre distribución ha sido ya portado a Linux y están empezando a aparecer multitud de aplicaciones comerciales. El hardware soportado es mucho mayor que en las primeras versiones del núcleo y abarca la práctica totalidad de arquitecturas y periféricos.

Linux es, pues, una reimplementación completamente gratuita de las especificaciones POSIX, con extensiones de SYSV y BSD (lo cual significa que parece UNIX pero no proviene del mismo código fuente base), que está disponible tanto en su versión fuente como ya compilada. El copyright pertenece a Linus B. Torvalds (Linus.Torvalds@.Helsinki.FI) y otros colaboradores, y es libremente redistribuible bajo las condiciones de la «GNU Public License».

2.2 Arquitectura de UNIX y LINUX

2.2.1 Arquitectura del Núcleo de Unix

Gráfica 96. Arquitectura del sistema UNIX



El núcleo del Sistema Operativo

El núcleo del sistema operativo Unix (llamado Kernel) es un programa escrito casi en su totalidad en lenguaje C, con excepción de una parte del manejo de interrupciones, expresada en el lenguaje ensamblador del procesador en el que opera.

Las funciones del núcleo son permitir la existencia de un ambiente en el que sea posible atender a varios usuarios y múltiples tareas en forma concurrente, repartiendo al procesador entre todos ellos, e intentando mantener en grado óptimo la atención individual.

El Kernel opera como asignador de recursos para cualquier proceso que necesite hacer uso de las facilidades de cómputo. Es el componente central de Unix y tiene las siguientes funciones:

- Creación de procesos, asignación de tiempos de atención y sincronización.
- Asignación de la atención del procesador a los procesos que lo requieren.
- Administración de espacio en el sistema de archivos, que incluye: acceso, protección y administración de usuarios; comunicación entre usuarios y entre procesos, y manipulación de E/S y administración de periféricos.
- Supervisión de la transmisión de datos entre la memoria principal y los dispositivos periféricos.

El Kernel reside siempre en la memoria central y tiene el control sobre la computadora, por lo que ningún otro proceso puede interrumpirlo; sólo pueden llamarlo para que proporcione algún servicio de los ya mencionados. Un proceso llama al Kernel mediante módulos especiales conocidos como llamadas al sistema.

El Kernel consta de dos partes principales: la sección de **control de procesos** y la de **control de dispositivos**.

La primera asigna recursos, programas, procesos y atiende sus requerimientos de servicio; la segunda, supervisa la transferencia de datos entre la memoria principal y los dispositivos periféricos.

Cuando se inicia la operación de la computadora, debe cargarse en la memoria una copia del núcleo, que reside en el disco magnético (operación denominada bootstrap). Para ello, se deben inicializar algunas interfaces básicas de hardware; entre ellas, el reloj que proporciona interrupciones periódicas. El Kernel también prepara algunas estructuras de datos que abarcan una sección de almacenamiento temporal para transferencia de información entre terminales y procesos, una sección para almacenamiento de descriptores de archivos y una variable que indica la cantidad de memoria principal.

A continuación, el Kernel inicializa un proceso especial, llamado proceso 0. En general, los procesos se crean mediante una llamada a una rutina del sistema (fork), que funciona por un mecanismo de duplicación de procesos. Sin embargo, esto no es suficiente para crear el primero de ellos, por lo que el Kernel asigna una estructura de datos y establece apuntadores a una sección especial de la memoria, llamada tabla de procesos, que contendrá los descriptores de cada uno de los procesos existentes en el sistema.

Después de haber creado el proceso 0, se hace una copia del mismo, con lo que se crea el proceso 1; éste muy pronto se encargará de "dar vida" al sistema completo, mediante la activación de otros procesos que también forman parte del núcleo. Es decir, se inicia una cadena de activaciones de procesos, entre los cuales destaca el conocido como despachador, o scheduler, que es el responsable de decidir cuál proceso se ejecutará y cuáles van a entrar o salir de la memoria central. A partir de ese momento se conoce el número 1 como proceso de inicialización del sistema, init. El proceso init es el responsable de establecer la estructura de procesos en Unix.

Normalmente, es capaz de crear al menos dos estructuras distintas de procesos: el modo monousuario y el multiusuario. Comienza activando el intérprete del lenguaje de control (Shell) en la terminal principal, o consola, del sistema y proporcionándole privilegios de "superusuario". En la modalidad de un solo usuario la consola permite iniciar una primera sesión, con privilegios especiales, e impide que las otras líneas de comunicación acepten iniciar sesiones nuevas. Esta modalidad se usa con frecuencia para revisar y reparar sistemas de archivos, realizar pruebas de funciones básicas del sistema y para otras actividades que requieren uso exclusivo de la computadora.

Init crea otro proceso, que espera pacientemente a que alguien entre en sesión en alguna línea de comunicación. Cuando esto sucede, realiza ajustes en el protocolo de la línea y ejecuta el programa login, que se encarga de atender inicialmente a los nuevos usuarios. Si la clave del usuario, y la contraseña proporcionadas son las correctas, entonces entra en operación el programa Shell, que en lo sucesivo se encargará de la atención normal del usuario que se dio de alta en esa terminal. A partir de ese momento el responsable de atender al usuario en esa terminal es el intérprete Shell.

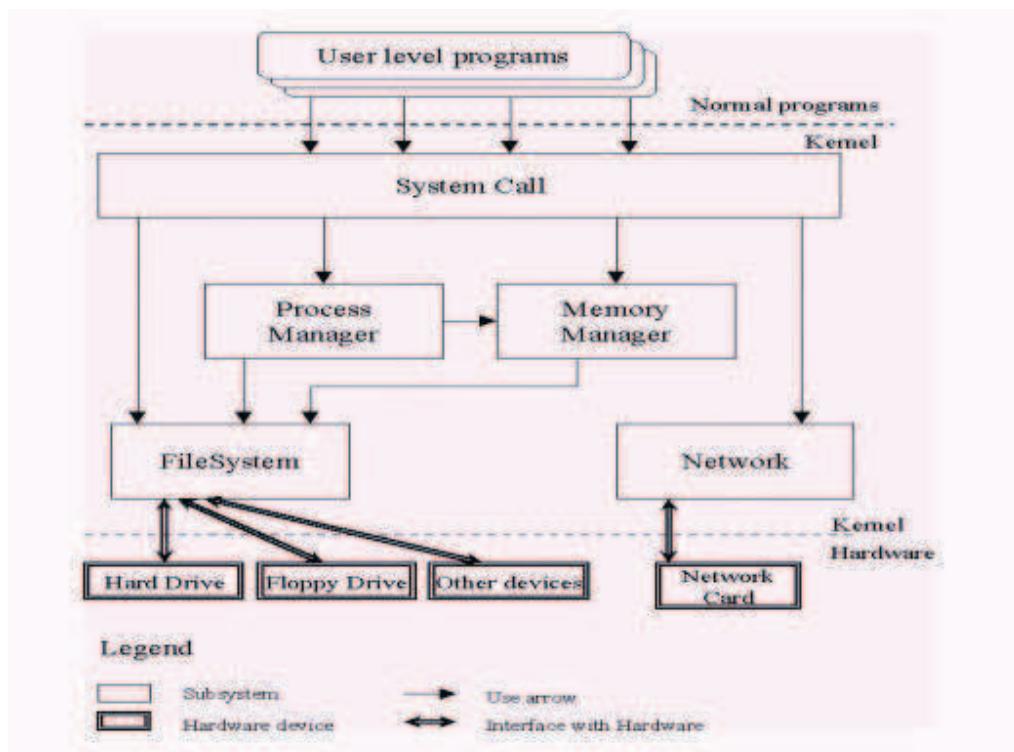
Cuando se desea terminar la sesión hay que desconectarse de Shell (y, por lo tanto, de Unix), mediante una secuencia especial de teclas (usualmente. < CTL > - D). A partir de ese momento la terminal queda disponible para atender a un nuevo usuario.

2.2.2 Arquitectura del Núcleo de Linux

Para empezar, la arquitectura abstracta de Linux puede describirse como la composición de un conjunto subsistemas o componentes interactuando unos con otros de forma simultanea. El núcleo se descompone en estos subsistemas basándose en su funcionalidad, donde cada subsistema pone su granito de arena propio y diferente en la labor del sistema operativo.

De esta manera, identificamos cinco subsistemas importantes en el núcleo, que son: el **subsistema de llamadas al sistema**, el **administrador de memoria**, el **administrador de procesos**, el **sistema de ficheros** y el **subsistema encargado de la Red**.

Gráfica 97. Arquitectura Sistema Operativo Linux



Esta arquitectura muestra todos estos subsistemas, escondiendo los mecanismos internos de cada uno de ellos.

La arquitectura de Linux es como un estilo "híbrido" o mezcla entre un *estilo de una arquitectura de capas* y un *estilo de una arquitectura orientada a objeto*.

Capas

La idea clave del *estilo de arquitectura por capas* es el planteamiento de una jerarquía de capas para los distintos módulos del núcleo, diferenciados según su funcionalidad y nivel de acceso a los dispositivos físicos. En Linux se pueden distinguir tres grandes capas, donde cada capa se comunica con los dispositivos pasando por las inmediatamente inferiores. Veamos un momento estas tres capas independientes:

- **1^a capa: Programas de usuario:** Aplicaciones ejecutadas por el usuario, programas desarrollados por él, rutinas del sistema operativo, etc.
- **2^a capa: Kernel:** Subsistemas encargados de facilitar las herramientas al programador que le permitan comunicarse con el hardware, sin tener que ocuparse del manejo detallado de cada uno de los dispositivos. Así, el programador se olvida de si pretende leer de una unidad de disquetera, de un disco duro, de un CDROM o de un recurso de red compartido, ya que dispone de una herramienta que "escribe en dispositivos" cuya tarea particular se reparten estos subsistemas para llevarla a cabo eficazmente.
- **3^a capa: Dispositivos hardware:** Son los propios aparatos, unidades de almacenamiento, dispositivos de lectura, impresoras, tarjetas de comunicaciones, sonido, imagen... etc.

Sin embargo, no se puede afirmar que Linux es estrictamente una arquitectura de capas, ya que las capas se forman de subsistemas que colaboran entre sí.

Si se revisa el otro estilo de arquitectura, el de arquitectura orientada a objeto, se tiene que los diferentes subsistemas "encapsulan" las operaciones que realizan sobre los dispositivos en forma de herramientas, de tal forma, que el programador hace uso de esas herramientas olvidándose de las vicisitudes del manejo del hardware y todo lo que ello conlleva.

Pero el estilo "híbrido" entre las dos ideas de arquitectura explicadas, es la que mejor describe el estilo de la arquitectura de Linux.

Subsistema de llamadas al sistema. El subsistema de llamadas al sistema es el que provee al núcleo de una interfaz externa. Es decir, es el subsistema que permite la comunicación a los programas de usuario interactuar con los demás subsistemas mediante lo que se denominan "llamadas al sistema".

Como cualquier interfaz, lo que se pretende es esconder los detalles del núcleo y del *hardware* del ordenador. De esta manera, se pueden añadir dispositivos o

simplemente cambiarlos sin que afecte al funcionamiento de los programas de usuario.

El conjunto de las llamadas al sistema pueden agruparse en categorías basándose en su funcionalidad, lo que nos llevaría a una descomposición de este subsistema en módulos. Pero no es de nuestro interés en la tarea que nos ocupa.

Subsistema Administrador de Memoria. El subsistema administrador de memoria, como su propio nombre indica, ofrece los servicios necesarios que permiten acceder a la memoria física del ordenador. Este subsistema esconde a los programas de usuario todo el manejo de la *memoria virtual*, que proporciona una mejor gestión de la memoria disponible del sistema, creando la apariencia de una mayor cantidad de memoria física utilizable, además de ofrecer la posibilidad de que los procesos comparten fragmentos de memoria.

El sistema de *memoria virtual* mejora en los siguientes aspectos:

- *Gran espacio de direcciones*: Parece como si el sistema dispusiera de una cantidad superior de la que realmente tiene, ya que la memoria virtual es mucho más grande que la memoria física del sistema.
- *Protección*: Cada proceso en el sistema tiene su propio espacio de direcciones virtual. Estos espacios son totalmente independientes, sin que puedan influir en otros procesos que corren al mismo tiempo. Además, existen herramientas del hardware que permiten proteger áreas de memoria contra escritura. Este mecanismo protege el código y los datos de ser sobrescritos por programas que acceden indebidamente a memoria.
- *Mapeado de la Memoria*: El mapeado de la memoria se utiliza para direccionar archivos en el espacio de direcciones de un proceso. De esta manera, el contenido de un archivo está vinculado directamente en el espacio de direcciones virtuales de un proceso.
- *Justa distribución de Memoria Física*: El administrador de memoria permite que cada proceso que se ejecute en el sistema tome una parte justa de la memoria física del sistema.
- *Memoria Virtual Compartida*: Aunque la memoria virtual permite que los procesos tengan espacios de direcciones separados, frecuentemente es útil para los procesos el hecho de trabajar sobre una misma porción de memoria, compartiendo datos de lectura y/o escritura. Además si dos programas iguales están corriendo al mismo tiempo, ambos podrían compartir, al menos, su porción de código.

El subsistema del Administrador de Memoria puede descomponerse en los módulos que desempeñan las distintas funciones. Y aunque estos módulos buscan objetivos independientes, deben cooperar para trabajar en el único modelo

estructurado de tablas de páginas, listas de páginas libres, listas de espacios de direcciones virtuales, etc.

Subsistema Administrador de procesos. Linux es un sistema operativo multiproceso y como tal permite que muchos procesos se ejecuten a la vez. Este subsistema gestiona la creación, planificación y eliminación de los procesos (de los que se encargan los tres módulos del subsistema). Todos los procesos tienen un espacio de direcciones virtuales propio y un conjunto de ficheros abiertos, así que para su gestión habrá que administrar todos estos recursos.

Para ello este subsistema trabaja con estructuras donde guarda de cada proceso que se está ejecutando, su identificador de proceso, su espacio de direcciones, sus ficheros abiertos, etc.

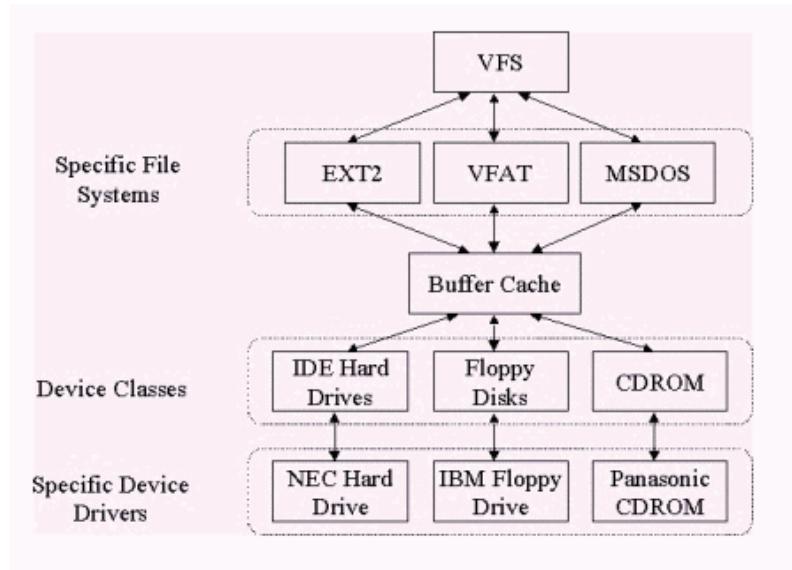
Subsistema del Sistema de ficheros. Linux utiliza el standard de sistema de ficheros de Unix. Este subsistema se encarga de todo acceso (ya sea lectura o escritura) a cualquier dispositivo físico utilizado para almacenar datos. Estos dispositivos físicos pueden ser discos duros (p. ej. IDE, SCSI), discos flexibles, CDROMS e incluso sistemas de ficheros de red (NFS). No solamente se soportan muchos dispositivos físicos, sino que además son varios los sistemas de ficheros que deben ser utilizados para poder acceder a los distintos dispositivos físicos.

Este subsistema se comunica con el resto de subsistemas mediante un *Sistema Virtual de Ficheros* (VFS), que actúa de interfaz. El VFS es un sistema de ficheros genérico que ofrece una estructura jerárquica y conceptual de archivos y directorios. El VFS sigue el modelo de sistemas de ficheros de Unix, construido mediante inodos, nexos simbólicos, etc.

La estructura del sistema de ficheros sigue el modelo de estructura por capas con niveles crecientes de abstracción.

La jerarquía comienza con los *drivers* específicos de cada dispositivo que se comunican directamente con el hardware. Más arriba tenemos los controladores de dispositivo según su tipo, los diferentes sistemas de ficheros y por último el interfaz ofrecido por el VFS.

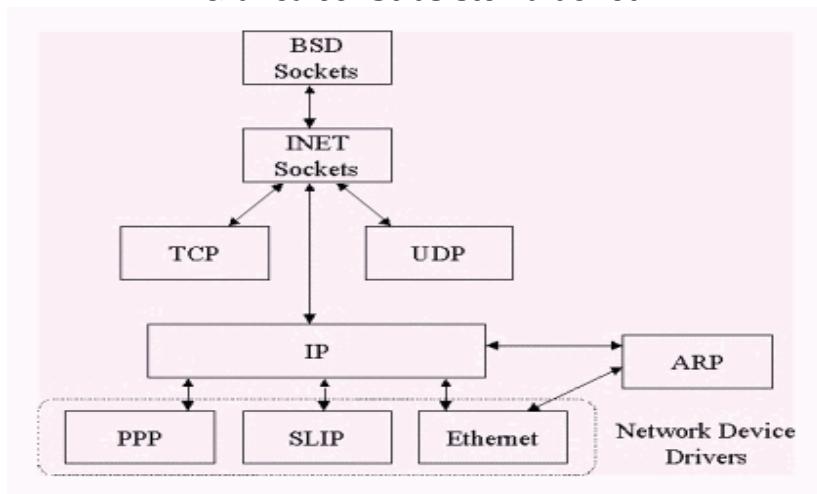
Gráfica 98. Subsistema de ficheros



Subsistema de Red. Linux provee servicios de red mediante la utilización de *sockets BSD* y los protocolos de red TCP/IP, para seguir en la línea de compatibilidad con Unix. De esta manera permite la comunicación con otras computadoras diferentes, que utilicen los mismos protocolos.

El subsistema de Red, parecido al sistema de ficheros, ya que también se comunica con el resto de subsistemas mediante una interfaz, que en este caso le proporciona los sockets BSD. Además también se estructura mediante capas, donde la capa más baja la ocupan los *drivers* de los dispositivos de red que mediante las capas del protocolo TCP/IP llegan a la interfaz de los sockets BSD, que se comunica con el núcleo, todo tal y como observamos en la figura.

Gráfica 99. Subsistema de red



2.3 Versiones y características Linux/Unix

2.3.1 Versiones

Existen numerosas versiones de UNIX para muchos sistemas, desde ordenadores personales hasta supercomputadores como el Cray Y-MP.

Entre las versiones de los sistemas operativos UNIX actuales cabe destacar:

- **Linux**, disponible para las familias x86 de Intel y compatibles, las estaciones Alpha de Digital (ahora Compaq-HP), la familia 68K de Motorola, estaciones MIPS, estaciones SP ARC, etc.
- **SunOS**, disponible para la familia 68K así como para la familia de estaciones de trabajo SP ARC de Sun Microsystems.
- **Solaris**, disponible para la familia SP ARC de Sun y la familia x86 de Intel y compatibles.
- **OSF1** (también conocido como OEC UNIX), para procesadores Alpha de Digital
- **Ultronix**, disponible para estaciones V AX de Digital.
- **SYSVR4** (también conocido como Novell-UNIX o UNIXware) disponible para las familias de procesadores x86, V AX.
- **IRIX**, disponible para MIPS.
- **AIX**, disponible para las series RS6000 de IBM y para el PowerPC.

Como puede observarse, se puede asegurar que existe una versión de UNIX para cualquier máquina medianamente conocida. Esto es debido a la gran portabilidad de su código, escrito en C, y a que éste es público y accesible.

2.3.2 Características de Linux/Unix

Debido a que LINUX es una versión de UNIX, ambos sistemas comparten la mayoría de las características técnicas y funcionalidades. Este capítulo se centrará en las características y funcionalidades de Linux, ya que tiene un carácter más didáctico y, al ser de libre distribución, el lector tendrá mayor facilidad para acceder a él. Además se comentarán, cuando las hubiere, las diferencias entre ambos sistemas operativos.

Linux es un sistema operativo completo con multitarea y multiusuario (como

cualquier otra versión de UNIX). Esto significa que pueden trabajar varios usuarios, simultáneamente en él y que cada uno de ellos puede tener varios programas en ejecución. Esto se consigue mediante la técnica de *time-sharing* (compartición de tiempo), en la que el sistema operativo, a través de su gestor de procesos, asigna intervalos de tiempo de CPU a cada proceso de la máquina.

El sistema Linux es compatible con ciertos estándares de UNIX a nivel de código fuente, incluyendo el IEEE POSIX.1, System V y BSD. Fue desarrollado buscando la portabilidad de los fuentes: casi todo el software gratuito desarrollado para UNIX se compila en Linux sin problemas. Y todo el software que se hace para Linux (código del núcleo, drivers, librerías y programas de usuario) es de libre distribución.

E] núcleo es capaz de emular por su cuenta las instrucciones del coprocesador 80387, con lo que en cualquier Intel 80386, con coprocesador matemático o sin él podrán ejecutar aplicaciones que requieran la presencia de la unidad de cálculo en coma flotante. Así, se evita que los programas tengan que hacer su propia emulación matemática.

Linux implementa el control de trabajos **POSIX** (que se usa en los shells **csh** y **bash**), las pseudoterminales (dispositivos pty) y teclados nacionales mediante manejadores de teclado cargables dinámicamente. Además, soporta consolas virtuales, lo que permite tener más de una sesión abierta en la consola de texto y conmutar entre ellas fácilmente.

Linux soporta diversos sistemas de ficheros para almacenar los datos. Algunos de ellos, como el **ext2fs** (un avanzado sistema de archivos propio con una capacidad de hasta 4 Terabytes y nombres de archivos de hasta 255 caracteres de longitud) han sido desarrollados específicamente para Linux. También soporta, entre otros, el sistema de ficheros **Minix-1** (el sistema de ficheros de Xenix), HPF8-2 del OS/2 2.1, el sistema de ficheros de MS-DOS (FAT) con el que se podrán acceder desde Linux a los disquetes y particiones en discos duros formateados con MS-DOS, también soporta la versión más moderna de este sistema de ficheros (FAT32) y todos los sistemas de archivo típicos de System V. Además, soporta el **ISO-9660**, que es el estándar seguido en el formato de los CD-ROMs.

Linux implementa todo lo necesario para trabajar en red con la pila de protocolos TCP/IP, desde manejadores para las tarjetas de red más populares hasta SLIP/PPP, que permiten acceder a una red TCP/IP por el puerto serie. Asimismo se implementan **PLIP** (para comunicarse por el puerto de la impresora) y **NFS** (**Network File System**, para acceso remoto a ficheros). Y también se han portado los clientes de TCP/IP, como FTP, telnet, NNTP y SMTP. Recientemente se han incorporado los protocolos de red Appletalk y Netware de Novell.

El núcleo de Linux ha sido desarrollado para utilizar las características del modo protegido de los microprocesadores 80386 80486 y superiores. En concreto, hace

uso de la gestión de memoria avanzada del modo protegido y otras características avanzadas. Cualquiera que conozca la programación del 386 en el modo protegido sabrá que este modo fue diseñado para su uso en UNIX (o tal vez Multics). Linux hace uso de esta funcionalidad precisamente.

El núcleo soporta ejecutables con paginación por demanda. Esto significa que sólo los segmentos del programa que se necesitan se cargan en memoria desde el disco. Las páginas de los ejecutables son compartidas mediante la técnica *copy on write* (copia en escritura), esto significa que varios procesos pueden usar la misma zona de memoria para ejecutarse. Cuando alguno intenta escribir en esa zona de memoria, la página (4Kb de memoria) se copia a otro lugar. Esta política de copia en escritura tiene dos beneficios: aumenta la velocidad y reduce la cantidad de memoria requerida para las aplicaciones.

Con el fin de incrementar la memoria disponible, Linux implementa paginación de memoria en disco: puede tener hasta 256 megabytes de espacio de intercambio o *swap* en el disco duro. Cuando el sistema necesite más memoria expulsará páginas inactivas al disco, permitiendo la ejecución de programas más grandes o aumentando el número de usuarios que puede atender a la vez. Sin embargo el espacio de intercambio no puede suplir totalmente a la memoria RAM, ya que el primero es mucho más lento que ésta.

La memoria dedicada a los programas y a la caché de disco está unificada. Por ello, si en cierto momento hay mucha memoria libre, el tamaño de la caché de disco aumentará acelerando así los accesos, de tal forma que toda la memoria libre puede ser usada para caché y este puede a su vez ser reducido cuando se ejecuten grandes programas.

Los ejecutables hacen uso de las librerías de enlace dinámico. Esto significa que los ejecutables comparten el código común de las librerías en un único fichero como sucede en SunOS. Así, los ejecutables serán más cortos a la hora de guardarlos en el disco, incluyendo aquellos que hagan uso de muchas funciones de librería. También pueden enlazarse estáticamente cuando se deseen ejecutables que no requieran la presencia de las librerías dinámicas en el sistema. El enlace dinámico se hace en tiempo de ejecución, con lo que el programador puede cambiar las librerías sin necesidad de recompilación de los ejecutables.

Se accede a los diferentes dispositivos como si se tratase de ficheros. De esta forma, es posible abstraerse del tipo de dispositivo con el que se está interactuando, ya que de cara al programador o al usuario se trabaja de forma idéntica con todos los dispositivos. Para el sistema operativo, cada dispositivo es un fichero (de hecho se podría decir que en UNIX todo es un fichero) y crea un fichero en su sistema de archivos por cada dispositivo instalado en la máquina. Cuando se quiere utilizar un dispositivo, se abre su fichero asociado y

se escribe o lee de él como si de un fichero de datos se tratara. El sistema operativo se encarga del resto.

A continuación, se exponen en una lista las principales características de Linux:

- Multitarea: varios programas (realmente procesos) se pueden ejecutar a mismo tiempo.
- Multiusuario: varios usuarios pueden trabajar en la misma máquina a mismo tiempo.
- Multiplataforma: corre en muchas CPUs distintas, no sólo Intel.
- Funciona en modo protegido 386.
- Protección de la memoria entre procesos, de manera que uno de ellos - pueda colgar el sistema.
- Carga de ejecutables por demanda: Linux sólo lee de disco aquellas partes de un programa que están siendo usadas actualmente.
- Política de copia en escritura (*copy on write*).
- Memoria virtual usando paginación (sin intercambio de procesos completos) a disco.
- La memoria se gestiona como un recurso unificado para los programas de usuario y para la caché de disco.
- Librerías compartidas de carga dinámica (DLL's).
- Casi totalmente compatible con POS IX, System V y BSD a nivel fuente.
- Código fuente disponible, incluyendo el núcleo completo y todos los drivers, las herramientas de desarrollo y todos los programas de usuario.
- Control de tareas POS IX.
- Pseudo-terminales (pty , s).
- Acceso a los diferentes dispositivos como si de ficheros se tratase.
- Emulación de la unidad de coma flotante 80387 en el núcleo.
- Consola virtual es múltiples: varias sesiones de login a través de la

consola.

- Soporte para varios sistemas de archivo comunes.
- Acceso transparente a particiones MS-DOS (o a particiones OS/2 FA T) mediante un sistema de archivos especial.
- Sistema de archivos ISO-9660 que lee todos los formatos estándar de CD-ROM.
- Pila de protocolos TCP/IP.
- Admite protocolos Appletalk.
- Software cliente y servidor NetWare disponible en los núcleos de desarrollo.

2.4 Instalación del sistema

Antes de proceder a la instalación del sistema operativo Linux en una máquina, se han de realizar dos pasos muy importantes:

- Verificar que la máquina cumple al menos con los requisitos mínimos de hardware que el sistema operativo a instalar requiere.
- Elegir la distribución a instalar entre las muchas existentes.

2.4.1 Requisitos hardware

Las necesidades *hardware* de Linux no son nada exigentes y mucho menos en la actualidad, donde cualquier ordenador del mercado supera, y con creces, los requisitos mínimos. Aún así, es conveniente tener presentes estos requisitos mínimos especialmente si se va a instalar en equipos de cierta antigüedad.

Estas necesidades varían en función del uso que se le vaya a dar a la máquina de la cantidad de usuarios a los que se vaya a dar servicio y a las aplicaciones que estos vayan a ejecutar.

Requisitos de placa base y de CPU

Actualmente Linux soporta, dentro de la arquitectura PC, sistemas con una CPU Intel 80386, 80486 o Pentium, Pentium II, III y IV. Esto incluye todas las variantes del tipo de CPU, como el 386SX, 486SX, 486DX, 486DX2, Pentium Pro. Pentium Celeron, Mendocino, etc. Los «clónicos» no Intel, como AMD y Cyrix también están soportados por Linux.

La placa base debe ser de arquitectura ISA o EISA en cuanto a bus se refiere.

El bus MicroChannel (MCA), que se encuentra en máquinas como los IBM/PS2, no está soportado actualmente.

Requisitos de memoria

Linux, comparado con otros sistemas operativos avanzados, necesita muy poca memoria para funcionar. Se debería contar con un mínimo de 4 Megabytes de RAM; sin embargo, es altamente recomendable tener al menos 32 Megabytes para un funcionamiento más fluido.

En equipos con poca memoria RAM, se reserva una parte del disco duro para espacio de intercambio o *swap* que se usa como memoria RAM virtual. Incluso si se dispone de bastante memoria RAM física en la máquina, es recomendable utilizar un área de *swap*. El área de *swap* no puede reemplazar a una memoria física RAM real pero puede permitir al sistema ejecutar aplicaciones cuyos requisitos de memoria superen a la RAM disponible en el sistema guardando en disco duro aquellas partes de código que están inactivas.

Una vez más, el uso final del equipo marca las necesidades de *hardware*. Si el sistema no va a ejecutar ningún tipo de entorno gráfico, con 16-32 ME habrá suficiente, pero si la intención es que disponga de alguno de ellos (posteriormente se verán más en detalle), es recomendable dotar al equipo con, al menos, 64 MB de RAM.

La controladora de disco duro

Si bien, no es imprescindible un disco duro para ejecutar Linux, ya que se puede ejecutar un sistema mínimo completamente desde disquete, esto resulta lento y muy limitado, por lo que la presencia de disco duro se hace obligada. Para ello, Linux soporta tanto las controladoras AT-estándar (16-bits) como las controladoras XT-estándar (8 bits). Soporta todas las controladoras MFM, RLL e IDE. La mayoría, pero no todas, las controladoras ESDI están soportadas (sólo aquellas que hacen emulación *hardware* de la ST506). La regla general para controladoras que no sean SCSI es que, si se puede acceder a las unidades desde MS-DOS u otro sistema operativo, se debe poder hacerlo desde Linux.

Linux también soporta un número de controladoras de disco SCSI, si bien el soporte para SCSI es más limitado a causa de la gran cantidad de estándares que existen para la interfaz de las controladoras. Las controladoras SCSI soportadas incluyen las Adaptec AHA1542B, AHA1542C, AHA1742A (versión de BIOS 1.34), AHA1522, AHA1740, AHA1740 (controladora SCSI-2, BIOS 1.34 en modo mejorado); Future Domain 1680, TMC-850, TMC-950; Seagate ST-02;

UltraStor SCSI; Western Digital WD7000FASST. Las controladoras clónicas basadas en estas tarjetas también deberían funcionar.

Espacio en disco

La cantidad de espacio en disco duro que se necesita depende, una vez más, en gran medida de la cantidad de servicios que se quieran ejecutar y de la cantidad de software que va a instalar. Linux es relativamente «pequeño» en relación a las implementaciones de UNIX. Se podría correr un sistema completo con sólo 20 *MB* de espacio en disco. Sin embargo, si se quiere disponer de espacio para expansiones y para paquetes más grandes como X- Window, se necesita más espacio.

Por otra parte, hay que calcular que, por cada usuario que vaya a utilizar la máquina, se tendrá que reservar espacio de disco para sus ficheros. A todo esto hay que añadir que se necesita crear un espacio de intercambio o *swap*, para ser usado como *RAM* virtual. (Esto último se verá más en profundidad al final de este apartado).

Así pues, antes de comenzar con la instalación del sistema operativo, se han de planificar cuidadosamente los requisitos de espacio en disco duro a tener de estos aspectos.

Linux soporta múltiples discos duros en la misma máquina, por lo que se puede disponer de espacio para Linux en múltiples unidades si es necesario. Es práctica recomendable separar en distintas particiones y, a ser posible, en distintos discos, las áreas de *swap*, sistema (*/root*) y usuario (*/home*) para una administración más sencilla.

Requisitos del adaptador de video

Linux soporta todas las tarjetas de vídeo estándar Hercules, CGA, EGA, VGA, IBM monocromo y Super VGA para la interfaz por defecto basada en texto. En general, si la combinación que tiene de monitor y tarjeta de vídeo funcionan bajo otro sistema operativo como MS-DOS, debería funcionar perfectamente con Linux. Las genuinas tarjetas CGA de IBM sufren el (d)efecto nieve bajo Linux, por lo que no es muy recomendable su uso.

Los entornos gráficos como el Sistema X-Window, KDE, Gnome, etc. tienen requerimientos propios de *hardware* para la tarjeta de vídeo que, generalmente, son más restrictivos que para el trabajo en modo comando, especialmente si se requiere una alta resolución de pantalla. Cada uno de ellos incluye una lista con el hardware soportado y probado, por lo que se debe comprobar en estos documentos si nuestro *hardware* está incluido.

Otro hardware

Para el resto del *hardware* de la máquina, teclado, ratón, impresoras, tarjeta de red, módems, unidades de cinta, unidades lectoras-regrabadoras de CD-ROM, DVD, escáneres, etc., se puede decir que siempre que el dispositivo siga los estándares internacionales estará soportado por el sistema. En cualquier caso, día a día se están añadiendo dispositivos a la lista de periféricos soportados, por lo que es conveniente informarse en las publicaciones de novedades y en los foros de desarrollo.

2.4.2 Distribuciones

En sí mismo, Linux es sólo el núcleo del sistema operativo, por lo que necesita acompañarse de otras aplicaciones para ser realmente útil; muchas de estas aplicaciones han sido portadas desde otros sistemas operativos (generalmente UNIX) o creadas específicamente para él. Todas estas aplicaciones están disponibles gratuitamente desde Internet pero, para evitar esta labor de búsqueda y descarga, algunas empresas presentan en paquetes colecciones de aplicaciones que, junto al núcleo, completan un sistema operativo completo.

A continuación se citan algunas de las más populares:

- **Slackware** (<http://www.slackware.com>). Una de las primeras y muy popular en un principio, actualmente ha perdido terreno debido a que su sistema de instalación no tiene control de versiones ni dependencias como el sistema RPM de Red Hat o el DEB de Debian.
- **Debian** (<http://www.debian.org>). Esta distribución no está desarrollada por ninguna compañía, sino que es fruto de la colaboración de la comunidad de Internet, por lo que se trata de una distribución totalmente gratuita donde todo el *software* es GNU/GPL y no incluye *software* comercial. Otra de sus ventajas es su sistema de instalación de paquetes DEB.
- **SUSE** (<http://www.suse.de>). Distribución que combina el sistema de paquetes de Red Hat (RPM) con una organización derivada de Slackware. Es la distribución más popular en Europa e incluye soporte para diferentes idiomas incluido el castellano. Es una de las más fáciles de instalar y configurar, además es muy completa en cuanto al número de paquetes se refiere.
- **Caldera** (<http://www.calderasystems.com>). Presenta la distribución OpenLinux, basada en el sistema de paquetes RPM de Red Hat y orientada a entornos comerciales, ya que incluye aplicaciones de oficina como Aplixware y Corel Word Perfecto.

- **Mandrake** (<http://www.Linux-mandrake.com>). Surge como un clon de RedHat integrando además el entorno gráfico KDE. Hoyes una de las distribuciones más extendidas.
- **Red Hat** (<http://www.redhat.com>). Es la distribución más famosa, precursora del sistema de instalación por paquetes RPM, muy sencilla de instalar gracias a su interfaz de instalación gráfica y con auto detección de dispositivos. A todo esto hay que añadir un excelente conjunto de aplicaciones comerciales en su distribución oficial. Debido a su popularidad es emulada por otras muchas como la Mandrake.

Una vez elegida y adquirida la distribución a instalar de Linux ya se puede comenzar con la instalación en sí pero antes es muy recomendable informarse de los dispositivos hardware que tiene instalada la máquina, ya que durante el proceso de instalación es muy posible que se deba suministrar esta información al asistente de la instalación.

Bajo UNIX y Linux, los dispositivos y las particiones tienen nombres muy distintos a los utilizados en otros sistemas operativos. Bajo MS-DOS, las disqueteras se identifican como A: y B:, mientras que las particiones del disco duro se identifican como C:, D:, etc. Bajo Linux, la denominación es algo diferente.

Los manejadores de dispositivos (en realidad son ficheros), que se encuentran en el directorio `/dev`, se usan para comunicar con los dispositivos del sistema (como discos duros o ratones).

Por ejemplo, si se tiene un ratón en el sistema, se puede acceder a él a través del manejador (fichero) `/dev/mouse`. Las disqueteras, discos duros y particiones tienen cada uno un manejador (fichero) propio. Es importante entender cómo son nombrados los dispositivos con el fin de poderlos usar.

Lista los nombres de diversos manejadores:

Primera disquetera (A:)	<code>/dev/fd0</code>
Segunda disquetera (B:)	<code>/dev/fd1</code>
Primer disco duro (todo el disco)	<code>/dev/hda</code>
Primer disco duro, partición primaria 1	<code>/dev/hda1</code>
Primer disco duro, partición primaria 2	<code>/dev/hda2</code>
Primer disco duro, partición primaria 3	<code>/dev/hda3</code>
Primer disco duro, partición primaria 4	<code>/dev/hda4</code>
Primer disco duro, partición lógica 1	<code>/dev/hda5</code>
Primer disco duro, partición lógica 2	<code>/dev/hda6</code>
..	
.	

Segundo disco duro (todo el disco)	/dev/hdb
Segundo disco duro, partición primaria 1	/dev/hdb1
..	
.	
Primer disco duro SCSI (todo el disco)	/dev/sda
Primer disco duro SCSI, partición primaria 1	/dev/sda1
..	
.	
Segundo disco duro SCSI (todo el disco)	/dev/sdb
Segundo disco duro, SCSI, partición primaria 1	/dev/sdb1

Es destacable que los discos duros SCSI se nombran de manera diferente a otros discos. Se accede a los IDE, MFM Y RLL a través de los dispositivos `/dev/hda`, `/dev/hdb`, etc. Las particiones de `/dev/hda` son `/dev/hda1`, `/dev/hda2`, etc. Sin embargo, los dispositivos SCSI se nombran con `/dev/sda`, `/dev/sdb`, etc., y las particiones con `/dev/sda1`, `/dev/sda2`..

EJEMPLO

Supongamos un ordenador que tiene un disco duro IDE con tres particiones primarias. Las dos primeras son para MS-DOS y la tercera es extendida y contiene dos particiones lógicas, ambas para ser usadas con Linux.

Los dispositivos quedarían representados con:

Primera partición MS-DOS (C:)	<code>/dev/hda1</code>
Segunda partición MS-DOS (D:)	<code>/dev/hda2</code>
Partición extendida	<code>/dev/hda3</code>
Primera partición lógica de Linux	<code>/dev/hda5</code>
Segunda partición lógica de Linux	<code>/dev/hda6</code>

Obsérvese que se ha saltado `/dev/hda4`, ya que corresponde a la cuarta partición primaria, que no existe en el ejemplo. Las particiones lógicas se nombran de forma consecutiva partiendo de `/dev/hda5`

A pesar de ser diferente en cada distribución de Linux, el método utilizado para instalar el *software* es, en general, como sigue:

1. **Reparticionar el o los discos duros.** Con el fin de reservar el espacio necesario para Linux. En este punto es muy conveniente ayudarse de alguna aplicación de gestión de discos como FIPS, Partition Magic o alguna otra utilidad de particionamiento que incluya la distribución (como por ejemplo, el Disk Wizard de *mandrake*).
2. **Arranque de la instalación de Linux.** Cada distribución de Linux incluye un método para arrancar inicialmente e instalar el software, usualmente un

disquete o un CD-ROM de arranque. Arrancando de esta forma, se entrará en un programa de instalación para el resto del software.

3. **Crear las particiones para Linux.** Después de reparticionar el disco, se deben crear particiones de Linux en dicho espacio. Esto se realiza con el programa fdisk:

Se debe crear, como mínimo, una partición para el software de Linux propiamente dicho y otra para el área de intercambio, aunque es recomendable crear varias particiones para el sistema de archivos de Linux con el fin de obtener una administración más fácil del sistema.

Para ello, hay que ejecutar el comando fdisk tecleando *fdisk <drive>* donde *<drive>* es el nombre de dispositivo con el que Linux identifica el disco duro donde quiere realizar las particiones (véase la Tabla 5.1). Por ejemplo, si se desea ejecutar fdisk sobre el primer disco SCSI del sistema, se utilizará el comando *fdisk /dev/sda*. Por defecto, fdisk actúa sobre */dev/hda* (el primer disco IDE). Para crear particiones de Linux en más de un disco, ejecute fdisk una vez por disco.

Se ilustra con un completo ejemplo ya que es quizás el punto más confuso de todo el proceso:

EJEMPLO 5.2

```
# fdisk /dev/hda
Command (m for help)
```

En este punto, fdisk está esperando un comando; se puede teclear m para obtener una lista de opciones.

```
Command (m for help): m
Command action
a toggle a bootable flag
d delete a partition
l list known partition types
m print this menu
n add a new partition
p print the partition table
q quit without saving changes
t change a partition's system id
u change display/entry units
v verify the partition table
w write table to disk and exit
x extra functionality (experts only)
```

Command (m for help):

El comando n se usa para crear una nueva partición. Para salir de fdisk sin salvar los cambios, se utiliza el comando q. Para salir escribiendo los cambios en la tabla de

particiones, se utiliza el comando `w`.

Lo primero que se debe hacer es mostrar su tabla de particiones actual y anotar sus datos, para referencias posteriores. Usar el comando `p` para esto.

```
Command (m for help): p
Disk /dev/hda: 16 heads, 38 sectors, 683 cylinders
Units = cylinders of 608 * 512 bytes
Device Boot Begin End Blocks Id System
/dev/hda1 * 1 1 203 61693 6 DOS 16-bit >=32M
Command (m for help):
```

En este ejemplo, se tiene una partición única en `/dev/hda1`, con 61.693 bloques (unos *60 MB* ya que en Linux un bloque son *1.024 bytes*).

Esta partición comienza en el cilindro 1 y finaliza en el 203. En total el disco tiene 683 cilindros de los cuales 480 están libres para crear particiones de Linux.

Para crear una nueva partición, se usa el comando `n`. En este ejemplo se crearán dos particiones primarias (`/dev/hda2` y `/dev/hda3`) para Linux.

```
Command (m for help): n
Command action
e extended
p primary partition (1-4)
p
```

Aquí, fdisk pide el tipo de partición a crear: extendida o primaria. En nuestro ejemplo se elige `p` pues solo se van a crear particiones primarias.

Partition number (1 – 4) :

Fdisk pregunta entonces por el número de la partición a crear; puesto que la 1 está en uso, la primera partición para Linux debe ser la 2.

Partition number (1 – 4) : 2
First cylinder (204 – 683) :

Ahora se debe introducir el cilindro de comienzo de la partición. Dado que actualmente no están en uso los cilindros 204 a 683, se escoge el primero disponible (204), ya que no hay razón para dejar huecos entre particiones.

First cylinder (204-683): 204
last cylinder or +size or +sizeM or +sizeK (204-683):

Ahora fdisk está preguntando acerca del tamaño de la partición a crear. Podemos especificar el cilindro de terminación de la partición o introducir directamente el tamaño requerido, en *bytes*, *kilobytes* o *megabytes*. Como se desea que la partición ocupe *80 MB*, se introduce `+80M`. Cuando se indica el tamaño de esta forma, fdisk lo redondea a un número de cilindros

Last cylinder or + size or + sizeM or +sizeK (204 – 683):

Ahora fdisk está preguntando acerca del tamaño de la partición a crear. Podemos especificar el cilindro de terminación de la partición o introducir directamente el tamaño requerido, en *bytes*, *kilobytes* o *megabytes*. Como se desea que la partición ocupe 80 *MB*, se introduce +80M. Cuando se indica el tamaño de esta forma, fdisk lo redondea a un número de cilindros.

```
Last cylinder or + size or + sizeM or +sizeK (204 – 683):  
+ 80M  
Warning: Linux cannot currently use 3390 sectors of this partition
```

Si se obtiene un mensaje como el anterior, este puede ignorarse. fdisk imprime este, aviso debido a que es un programa antiguo que data de cuando las particiones de Linux no podían superar los 64 *MB*.

Ahora se va a crear la segunda partición. Para el ejemplo, se va a crear de 10 MB.

```
Command (m for help): n  
Command action  
e extended  
p primary partition (1-4)  
p  
Partition number (1-4): 3  
First cylinder (474-683): 474  
Last cylinder or + size or + sizeM or +sizeK (474 – 683):  
+ 10M.
```

Finalmente, se ha de ver la tabla de particiones, para comprobar el estado final del disco. Una vez más, es recomendable anotar la información que se presenta, sobre todo los tamaños en bloques de las nuevas particiones. Esta información será útil al crear, más tarde, los sistemas de ficheros. Además, se debe verificar que las particiones no se solapen.

```
Command (mfor help): p  
Disk /dev/hda: 16 heads, 38 sectors, 683 cylinders  
Units = cylinders of 608 * 512 bytes  
Device Boot Begin End Blocks Id System  
/dev/hda1 * 1 1 203 61693 6 DOS 16 - bit >= 32M  
/dev/hda2 204 204 473 82080 81 Linux/MINIX  
/dev/hda3 474 474 507 10336 81 Linux/MINIX
```

Se puede ver que ahora en /dev/hda2 hay una partición de 82.080 bloques (aproximadamente 80 *MB*) y en /dev/hda3 hay 10.336 bloques (unos 10*MB*). Los cilindros sobrantes (508 a 683) se quedan sin usar. Puede que se deseé hacerlo así - para más adelante crear más particiones. Finalmente, se utilizará el comando w para escribir los cambios en el disco y salir.

```
Command (m for help): w  
#
```

Durante el proceso de creación de particiones, o posteriormente a él, se debe indicar el tipo que tendrá cada partición, para ello se utiliza el comando t en fdisk.

Así, por ejemplo, para la partición dedicada a área de intercambio se selecciona el tipo *Linux swap*, normalmente indicado con el número 82. Para la partición que albergará el sistema de archivos de Linux se selecciona el tipo *Linux native*. Se puede usar el comando L para ver una lista de los tipos de particiones conocidas, y luego t para establecer el tipo de la partición. De esta forma, el *software* de instalación podrá encontrar automáticamente las particiones creadas y sugerir las más apropiadas durante el proceso de instalación.

Ningún cambio realizado durante la ejecución de fdisk tendrá efecto hasta que se teclee el comando w, por lo que se puede jugar con diferentes configuraciones y salvar los cambios sólo cuando estos sean correctos totalmente. Además, se puede usar el comando q para abandonar fdisk sin hacer ningún cambio. Es importante recordar que las particiones de otros sistemas operativos no deben tocarse desde el programa fdisk de Linux.

Asimismo, hay que recordar que no se puede arrancar Linux desde una partición que comience más allá del cilindro 1.023. Por lo tanto, se puede crear la partición de raíz en el rango inferior a este cilindro o, si esto es imposible, arrancar siempre desde un disquete.

Algunas distribuciones de Linux necesitan rearrancar el sistema tras ejecutar fdisk. Esto permite que los cambios en la tabla de particiones tengan efecto. Las nuevas versiones de fdisk cambian de forma automática esta información en el núcleo, con lo que no es necesario rearrancar.

4. Crear los sistemas de ficheros y el espacio de intercambio. En este momento, se debe crear uno o más sistemas de ficheros, que se utilizarán para guardar los archivos, en las particiones recién creadas. Además, si se piensa usar espacio de intercambio (altamente recomendable), se debe crear dicho espacio en una de las particiones para Linux.

Antes de que se puedan usar las particiones de Linux (creadas en el punto anterior) para almacenar ficheros, hay que crear los sistemas de ficheros en ellas. La creación de un sistema de ficheros es análoga a formatear una partición de MS-DOS u otros sistemas operativos.

Hay varios tipos de sistemas de ficheros disponibles en Linux. Cada tipo de sistema de ficheros tiene su propio formato y características (como longitud del nombre de los ficheros, tamaño máximo, etc.). Además, Linux soporta sistemas de ficheros «de terceros» como el de MS-DOS.

El tipo de sistema de ficheros más usado en Linux es el Sistema de Ficheros Extendido (ext2fs). Es uno de los más eficientes y flexibles sistemas; permite hasta 256 caracteres en los nombres de los ficheros y tamaños de estos de hasta 4 TB.

Actualmente, las distribuciones, en el propio procedimiento de instalación, crean los sistemas de ficheros de forma automática. Aún así es conveniente saber cómo se realiza esta tarea manualmente.

Para crear un sistema tipo ext2fs se utiliza el comando:

```
mke2fs -c <partition> <size>
```

donde *<partition>* es el nombre de la partición y *<size>* es el tamaño de la partición en bloques.

Por ejemplo, para crear un sistema de 82.080 bloques en */dev/hda2*, se usa el comando:

```
# mke2fs -c /dev/hda2 82080
```

Si se desea usar varios sistemas de ficheros en Linux, se necesitará repetir el comando mke2fs por cada sistema de ficheros.

Área de intercambio

Como ya se ha explicado anteriormente, Linux permite utilizar memoria de disco para emular a la RAM física mediante el sistema de paginación a disco. De esta forma se logra poder ejecutar un mayor número de procesos simultáneamente o ejecutar procesos con grandes requisitos de memoria que de otra forma no podrían ser ejecutados.

Es recomendable reservar un tamaño de intercambio igual al doble de la memoria RAM del equipo para optimizar el mecanismo de paginación a disco. Así pues, si el equipo en el que se va a instalar el sistema operativo cuenta con 64 MB de RAM, se reservarán 128 ME de espacio de intercambio.

El comando utilizado para preparar una partición de intercambio es mkswap, tecleándose:

```
# mkswap -c <partition> <size>
```

donde *<partition>* es el nombre de la partición de swap y *<size>* es el tamaño de la partición, en bloques. Por ejemplo, si la partición de intercambio es la */dev/hda3* y tiene 10.336 bloques, se ha de teclear el comando:

```
# mkswap -c /dev/hda3 10336
```

La opción -c indica a mkswap que compruebe si hay bloques erróneos en la partición mientras la crea.

Si se usan varias particiones de intercambio, se necesitará ejecutar el comando mkswap apropiado para cada partición.

Después de preparar el área de swap, hay que decirle al sistema que la use aunque, normalmente, el sistema comienza a usarla automáticamente durante el arranque. Sin embargo, durante la instalación como aún no tiene instalado el *software* de Linux, se tiene que activar a mano.

El comando para hacerlo es swapon y. tiene el formato *swapon <partition>*.

En el ejemplo anterior, para activar el espacio de intercambio en /dev/hda3, se usará el comando:

```
# swapon /dev/hda3
```

5. **Instalar los programas en los sistemas de ficheros.** Ya sólo resta instalar el *software* en la máquina. Este proceso está bastante automatizado por parte de las distintas distribuciones y, generalmente, se presenta un menú en el que se muestran los paquetes de aplicaciones, con una breve descripción de su funcionalidad, incluidos en la distribución para que el usuario elija los que deseé.

El resto del proceso consiste simplemente en la copia a disco de dichos paquetes.

La mayoría de las distribuciones de Linux proporcionan un programa de instalación que asesora en cada paso de la instalación y automatiza, en mayor o menor medida, algunos o todos de estos pasos.

2.5 La interfaz de usuario

2.5.1 El sistema X - Window

X-Window es el entorno gráfico habitual de los sistemas UNIX. El sistema X-Window se compone de dos partes principales: el servidor X y el programa para la gestión de las ventanas o gestor de ventanas. **El servidor X** es el programa que se encarga realmente de dibujar en la pantalla. Por el contrario, el **gestor de ventanas**, como su nombre indica, es el encargado de crear las ventanas y gestionar su apariencia. Debido a este modelo de trabajo (cliente-servidor), la apariencia de las aplicaciones varía según se use uno u otro gestor de ventanas, entre los que destacan por su sencillez de uso y bajos requisitos hardware los entornos **twm**, **fvwm**, **olvwm**, etc. y, por su versatilidad y riqueza gráfica, **GNOME** y **KDE**.

Al instalar Linux, el sistema puede preguntar si se desea arrancar Linux en modo texto o en modo gráfico. Si se ha seleccionado esta última opción, Linux arrancará directamente X-Window. En caso contrario, en la línea de comandos hay que escribir *startx* con lo cual se arranca el modo gráfico. Por defecto, se ejecutará el entorno gráfico GNOME (en distribuciones que no sean Red Hat 6.0, se puede arrancar por defecto otro entorno como KDE o AfterStep).

El sistema X-Window es un interface gráfico estándar para máquinas UNIX. Es un potente entorno que soporta muchas aplicaciones. Usando X-Window, el usuario puede tener múltiples terminales a la vez en ventanas sobre la pantalla, cada una conteniendo una sesión diferente.

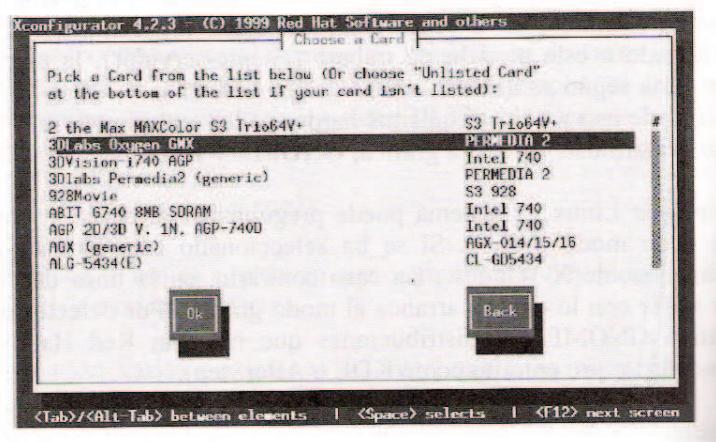
Han sido escritas muchas aplicaciones específicamente para X, como juegos, utilidades gráficas, herramientas de programación, documentación y un largísimo etcétera.

Con Linux y X, el sistema es una auténtica estación de trabajo. Junto con una red *TCP/IP* se puede incluso visualizar aplicaciones que se están ejecutando en otras máquinas en su pantalla local, tal y como es posible con otros sistemas que ejecuten X.

El sistema X-Window fue desarrollado originalmente en el MIT y es de libre distribución. A pesar de esto, muchas empresas han distribuido sus mejoras particulares al diseño original de X- Window. La versión de X- Window disponible para Linux es conocida como XFree86 (una adaptación de XIIR56 de libre distribución para sistemas UNJX basados en 80386, como es Linux). Linux dispone en la actualidad de la versión XIIR6 de X-Window.

XFree86 soporta una gran variedad de tarjetas de video, incluyendo VGA. Súper VGA y gran cantidad de tarjetas aceleradoras de video. Esta es una distribución completa de X-Window que contiene el servidor de X, muchas aplicaciones ~ utilidades, librerías de programación y documentación.

Aplicaciones X estándar incluyen xterm (emulador de terminal usado por la mayoría de las aplicaciones en modo texto dentro de X), xdm (el gestor de sesiones.. maneja los login), xclock (un reloj simple), xman (un lector de páginas de manua: para X) y muchos más. El número de aplicaciones disponibles para X Windows en Linux es demasiado numeroso como para mencionarlas aquí, pero la distribución básica de XFree86 incluye las aplicaciones estándar que se encontraban en la versión:, original del MIT. Muchas otras están disponibles de forma separada y, teóricamente cualquier aplicación escrita para X Windows debería compilar limpiamente bajo Linux.



Ventana de Configuración de XConfigurator

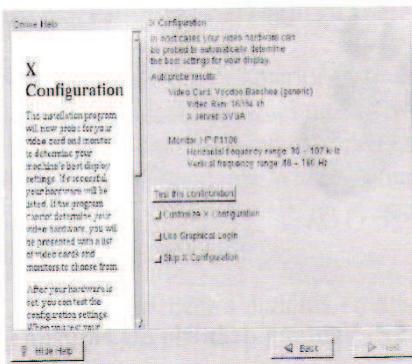
La distribución estándar XFree86 incluye **twm** (el clásico controlador z ventanas del MIT) aunque hay disponibles controladores mucho más avanzados como **Open Look Virtual Window Manager (olvwm)**.

Para configurar en servidor X, se pueden usar las siguientes utilidades:

- **Xf86config.** Programa en modo texto y modo consola, actualmente en desuso, pero igualmente útil.
- **Xconfigurator.** Programa en modo consola, pero con ventanas (figura 5.2).
- **XF86Setup.** Programa en modo gráfico VGA.
- **X Configuration.** Programa en modo gráfico (ver figura 5.3).

Durante la configuración, se especifica el modelo de la tarjeta gráfica instalada y del monitor, así como las resoluciones y frecuencias que estos dispositivos soportan, de entre los que se elige una combinación como predeterminada. También se indican los directorios en los que se guardará la configuración así como el gestor de ventanas elegido entre los disponibles en el sistema.

Un controlador de ventanas muy popular entre los usuarios de Linux es el **fvwm**. Es un pequeño controlador que requiere menos de la mitad de la memoria usada por twm. Proporciona aspecto de 3D para las ventanas, así como un escritorio virtual: si el usuario desplaza el ratón al borde de la pantalla, la pantalla entera es desplazada, pareciendo mucho más grande de 10 que realmente es.



Panel de configuración de X-Window

El controlador de ventanas fvwm es altamente configurable y permite el acceso a todas las funciones (tanto desde el teclado como desde el ratón). Muchas distribuciones de Linux 10 utilizan como controlador de ventanas estándar.

La distribución XFree86 contiene librerías para programación y ficheros de cabecera para aquellos programadores mañosos que deseen desarrollar aplicaciones para X. Están soportados varios conjuntos de controles como los de Athena, Open Look y Xaw3D. Todas las fuentes estándar, mapas de bits, páginas de manual y documentación están incluidas. PEX (interface de programación para gráficos 3D) también está soportado.

Muchos programadores de aplicaciones para X usan el conjunto comercial de controles **Motif** para el desarrollo. Algunos vendedores proporcionan licencias simples o multiusuario de una versión ejecutable de Motif para Linux, pero, como Motif en sí es bastante caro, no disponen de él demasiados usuarios de Linux. A pesar de todo, los ejecutables estáticamente enlazados con las librerías de Motif pueden ser distribuidos libremente. Por lo tanto, si se escribe un programa usando Motif y se desea distribuirlo libremente, se deberá proporcionar un ejecutable con las librerías enlazadas estáticamente para que los usuarios que no posean Motif puedan utilizarlo.

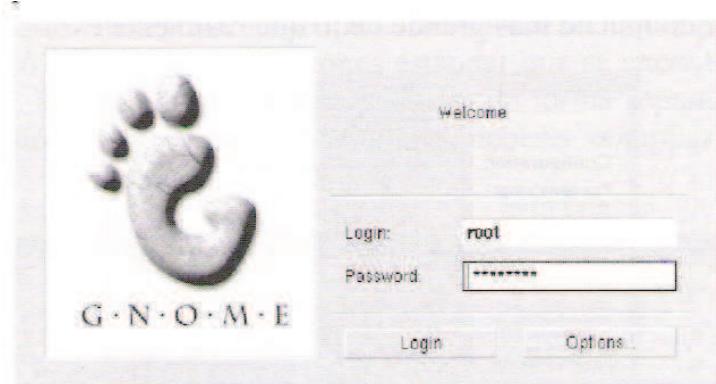
2.5.2 GNOME

GNOME es un entorno de ventanas muy popular debido a que está incluido en la distribución Red Hat. Se trata de un *software* gratuito ya que se distribuye bajo los términos de la licencia GNU.

GNOME (GNU Network Object Model Environment) está basado en el entorno GTK y no se limita a ser un simple manejador de ventanas, sino que es un completo escritorio interactivo que proporciona una interfaz de usuario potente, consistente y altamente modular.

Cuando Linux arranca en modo gráfico y el gestor de ventanas seleccionado es el

GNOME, aparece una ventana similar a la siguiente:

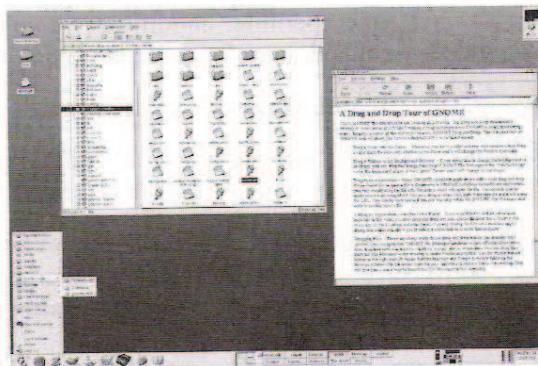


En ella se debe introducir el nombre del usuario y la clave (tanto Linux como UNIX distinguen entre letras mayúsculas y minúsculas, por lo que *root* no es mismo que *Root* o *ROOT*).

Cuando se sale del sistema, volverá a aparecer esta misma ventana. Para apagar el ordenador se puede seleccionar el botón de *Options...*, tras lo que aparece un menú con las siguientes opciones:

- **Sessions.** Permite elegir al usuario el entorno de ventanas con el que va a trabajar, los entornos más habituales son: AnotherLevel, Default (arranca el entorno por defecto instalado que puede ser cualquiera de los otros), FailSafe (Modo a prueba de fallos), Gnome o KDE
- **Language.** Permite cambiar el idioma en el que se muestran algunos de los mensajes del sistema.
- **System.** Contiene dos opciones: *Reboot* (para rearrancar el sistema) y *Halt* para apagarlo. Nota: NUNCA se debe apagar directamente el ordenador siempre hay que apagarlo empleando la opción *halt* anterior (tras salir del entorno con *logout*) o empleando el comando *halt* en el caso de trabajar en modo texto.

Tras introducir el nombre del usuario y el *password* aparecerá una pantalla similar a la siguiente:



Entorno de Trabajo GNOME

Como puede observarse en la figura anterior, este entorno es muy similar a otros como Windows, OS/2 o Macintosh. Al igual que estos entornos, GNOME está diseñado para ser empleado con el ratón e incluye elementos comunes con estos entornos como íconos, menús, etc. Al igual que Windows, incluye en la parte inferior una barra, el Panel de GNOME (GNOME Panel), en el cual se encuentran accesos directos a determinados programas de uso común.

La pantalla de GNOME se encuentra dividida en dos zonas principales como se puede apreciar en la figura anterior: la parte superior en la que aparecen las ventanas de las aplicaciones y documentos del usuario recibe el nombre de escritorio, y la parte inferior de la pantalla que recibe el nombre de panel de GNOME. Este elemento está diseñado para contener la siguiente información:

- La huella de pie o *footprint*. Al seleccionar este elemento aparece un menú similar al menú Inicio de Windows 9x con las aplicaciones más importantes instaladas.
- Las *applets* que son enlaces a las aplicaciones de uso más frecuente como la consola, Netscape, la ayuda, etc.
- El acceso a los escritorios virtuales. Al contrario que en Windows, X Window permite organizar las ventanas en varios escritorios virtuales.
- GNOME dispone también de un área específica en la que aparecen los botones representativos de las ventanas activas en el sistema.
- En los dos extremos del panel aparecen dos botones con flechas con los que el usuario puede colapsar el panel de forma que se maximice el área útil del escritorio. Haciendo clic con el botón derecho del ratón sobre cualquiera de los elementos anteriores, aparecerá un menú contextual que permite configurar el elemento.

Las cuatro opciones más habituales son:

- *Remove from Panel* que permite eliminar el *applet* del panel.
- *Move applet* que permite modificar la posición del *applet* arrastrándola y soltándola en la nueva posición.
- *About* que muestra información sobre el autor del *applet*.
- *Properties* que abre un cuadro de diálogo en el que se permite personalizar todas las características del *applet*.

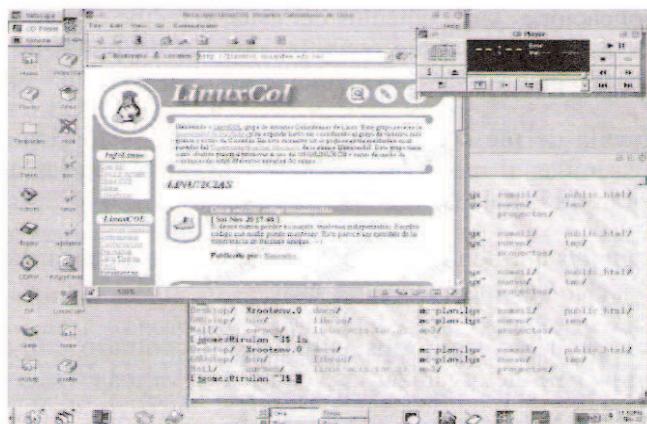
Seleccionado así mismo en alguna de las áreas libres del panel se despliega otro menú que permite configurar todo el panel, añadir y quitar *applets* del mismo. A este mismo menú se puede acceder a través de la opción Panel del menú principal: (*footprint*)

De igual forma que en el panel, en el escritorio también se puede hacer clic con el botón derecho, lo que despliega un menú contextual con diversas opciones.

Para salir de GNOME hay que seleccionar la opción *Log Out* que aparece en la parte inferior del menú.

2.5.3 KDE

KDE (K Desktop Environment) está basado en el entorno Qt. KDE es uno de los entornos gráficos más populares de Linux puesto que une una gran facilidad de uso a un entorno bonito y agradable. Al arrancar KDE aparece el escritorio en el que se pueden encontrar elementos similares a los de otros entornos:



Entorno de Trabajo en KDE

Por defecto la pantalla de KDE se divide en tres partes fundamentales:

- Panel de KDE
- Escritorio
- Panel de ventanas

El panel de KDE contiene accesos directos a las aplicaciones más empleadas así como dos menús:

- El equivalente al menú Inicio de Windows. Es el menú a través del cual se pueden ejecutar las aplicaciones. Al seleccionar este elemento, se despliega un menú subdividido en distintas categorías. KDE incluye una gran cantidad de utilidades que se integran con el entorno.
- El segundo menú del KDE. En el menú de ventanas se puede acceder a todas las que estén abiertas en los distintos escritorios. Al contrario que otros entornos gráficos, X Window permite organizar las ventanas en distintos escritorios virtuales. Para cambiar de escritorio virtual se puede escoger uno de los cuatro botones que aparecen en el panel.

Justo encima del panel de KDE, aparece el escritorio; al igual que en Windows este elemento contiene iconos que permiten acceder a los elementos más comunes como las unidades de disco o la papelera.

Por último, en la parte superior del escritorio aparece otra barra, en la que irán apareciendo botones por cada ventana que se cree.

Las ventanas en el KDE tienen un aspecto similar al de las ventanas de Windows (al menos con el aspecto básico), pudiéndose distinguir diversas partes:

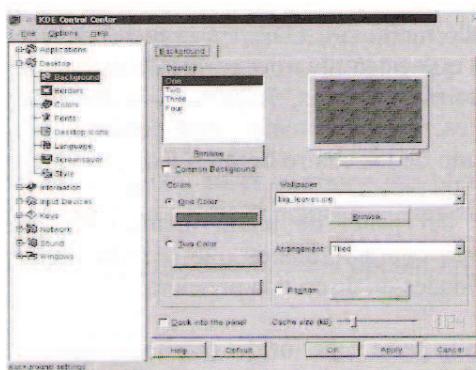
- En la parte superior izquierda, aparece el ícono de la aplicación que, al seleccionarlo, mostrará un menú con las opciones básicas de manipulación de ventanas: minimizar, maximizar, cerrar; así como otras no tan habituales como enviar la ventana a otros escritorios.
- En la parte superior central de la ventana, se encuentra la barra de títulos de la ventana.
- En la parte superior derecha aparecen tres botones con las opciones de minimizar, maximizar y cerrar.

Esta es la disposición por defecto pero puede ser adaptada a los gustos del usuario de una forma muy sencilla. Por debajo de este elemento se extiende la barra de menús y de herramientas y el área de trabajo de la aplicación.

KDE Control Center

Esta aplicación es la principal encargada de configurar KDE y a ella se puede acceder de muchas formas, tanto desde el ícono que aparece en el panel, como desde cualquiera de las entradas al menú *Settings*, en cuyo caso sólo se accede a una de las posibles opciones de configuración. Cuando se arranca, aparecerá una ventana dividida en dos:

- En la parte de la izquierda aparecen ordenadas las diferentes categorías de configuración (que coinciden con las categorías del menú *Settings*).
- En la parte derecha se abrirán los distintos cuadros de diálogo que permiten configurar KDE (por ejemplo, seleccionando la opción *Desktop/Background*, aparecerá a la derecha el cuadro de diálogo que permite cambiar la imagen de fondo de los escritorios virtuales).



KDE Control Center

2.5.4 EL SHELL

Hasta este momento, se han visto algunos de los entornos gráficos más importantes existentes para Linux. No obstante, cualquier usuario de Linux acabará antes o después relacionándose con el sistema empleando el modo texto.

Este modo se basa en la ejecución de una serie de comandos, que son interpretados por un programa o shell.

El kernel o núcleo del sistema ofrece sus servicios y el acceso a dispositivos través de llamadas al sistema y llamadas de funciones. Pero, mientras los programas acceden directamente a estos, para que el usuario pueda acceder a los servicios de sistema se necesita un programa que haga de intermediario entre el usuario (a través del terminal) y el *kernel*. Este programa es el *shell* y, como su nombre indica, es algo que envuelve y protege al núcleo como si este fuese una perla.

Linux dispone de varios de estos programas pero el más habitual es conocido como bash o Bourne Shell. Alternativas a este son el sencillo sh, el csh (C shell), tcsh, el ksh de Korn, etc.

Bajo Linux hay algunas diferencias en los intérpretes de comandos disponibles. Dos de los más usados son los ya mencionados **Bourne Again Shell o Bash** (*/bin/bash*). Bash es un equivalente al Bourne con muchas características avanzadas de la C shell. Como Bash es un superconjunto de la sintaxis del Bourne, cualquier guión escrito para el intérprete de comandos Bourne Standard funcionará en Bash. Si se prefiere el uso del intérprete de comandos C, Linux tiene el Tcsh, que es una versión extendida del C original.

El tipo de intérprete de comandos a usar es una cuestión de gustos. Algunas personas prefieren la sintaxis del Bourne con las características avanzadas que proporciona Bash y otras prefieren el más estructurado intérprete de comandos C. En lo que respecta a los comandos usuales como cp, ls, Etc, es indiferente el tipo de intérprete de comandos usado, la sintaxis es la misma. Sólo cuando se escriben scripts (guiones) para el intérprete de comandos o se usan características avanzadas aparecen las diferencias entre los diferentes intérpretes de comandos.

Si Linux se ha arrancado en modo texto, el sistema arranca de forma directa el shell y queda a la espera de introducción de nuevos comandos. Si se ha arrancado en modo gráfico se puede acceder al shell de dos formas:

➤ **Presionando alguna de las siguientes combinaciones de teclas:**

```
<ctrl>+<alt>+<F1>
<ctrl>+<alt>+<F2>
<ctrl>+<alt>+<F3>
<ctrl>+<alt>+<F4>
<ctrl>+<alt>+<F5>
<ctrl>+<alt>+<F6>
```

Esto hace que el sistema salga del modo gráfico y acceda a alguna de las seis consolas virtuales de Linux, a las cuales también se puede acceder cuando se arranca en modo de texto. Para volver al modo gráfico hay que presionar <ctrl>+<alt>+<F7> o <ctrl>+<alt>+<F8>.

➤ La segunda forma es acceder al *shell* desde el mismo entorno gráfico. Para esto hay que abrir un programa llamado terminal o consola, por ejemplo: kconsole (en el entorno KDE), xterm, gnome-terminal (en GNOME), etc.

En su forma más habitual (los *shells* de Bourne o de Korn), el sistema operativo

utiliza un signo de \$ como prompt para indicar que está preparado para aceptar comandos, aunque este carácter puede ser fácilmente sustituido por otro u otros elegidos por el usuario. En el caso de que el usuario acceda como administrador este signo se sustituye por #.

En este punto, el usuario puede teclear comandos con o sin argumentos que serán ejecutados por el sistema operativo.

Como utilidad práctica, cabe mencionar que, cuando sea necesario introducir el nombre de un fichero o directorio como argumento a un comando, Linux permite escribir las primeras letras del mismo y realiza un autorrellenado al presionar la tecla del tabulador. Si no puede distinguir entre diversos casos, llenará hasta el punto en el que se diferencien.

2.6 Estructura de archivos

2.6.1 Tipos de archivos

La base del sistema de archivos de Linux es, obviamente, el archivo que no es otra cosa que la estructura empleada por el sistema operativo para almacenar información en un dispositivo físico como un disco duro, un disquete, un CD-ROM, etc. Como es natural, un archivo puede contener cualquier tipo de información, desde una imagen en formato GIF o JPEG a un texto o una página WEB en formato HTML. El sistema de archivos es la estructura que permite que Linux maneje los archivos que contiene.

Todos los archivos de Linux tienen un nombre, el cual debe cumplir unas ciertas reglas:

- Un nombre de archivo puede tener entre 1 y 255 caracteres.
- Se puede utilizar cualquier carácter excepto la barra inclinada / y no es recomendable emplear los caracteres con significado especial en Linux, que son los siguientes: «=», «1|», «-», «'», «"», «'», «*», «;», «->, «?», «[», «]», «(», «)>, «!», «&», «->, «<» y «>>>. Para emplear ficheros con estos caracteres o espacios hay que introducir el nombre del fichero entre comillas.
- Se pueden utilizar números exclusivamente si así se desea. Las letras mayúsculas y minúsculas se consideran diferentes y, por lo tanto, no es lo mismo carta.txt que Carta.txt o carta.Txt

Como en Windows, se puede emplear un cierto criterio de «tipo» para marcar las distintas clases de ficheros empleando una serie de caracteres al final del nombre

que indiquen el tipo de fichero del que se trata. Así, los ficheros de texto, HTML, las imágenes PNG o JPEG tienen extensiones .txt, .htm (o .html), .png y .jpg (o jpeg) respectivamente.

Pese a esto Linux sólo distingue tres tipos de archivos:

- Archivos o ficheros ordinarios que son los mencionados anteriormente.
- Directorio (o carpetas). Son archivos especiales que agrupan otros ficheros de una forma estructurada.
- Archivos especiales. Son la base sobre la que asienta Linux, puesto que representan los dispositivos conectados a un ordenador, como puede ser una impresora. De esta forma, introducir información en ese archivo equivale a enviar información a la impresora. Para el usuario estos dispositivos tienen el mismo aspecto y uso que los archivos ordinarios.

2.6.2 Enlaces

Los enlaces son un tipo de archivo ordinario cuyo objetivo es crear un nuevo nombre para un archivo determinado. Una vez creado el enlace simbólico este permite acceder al fichero que enlaza de igual modo como si se accediese desde el archivo original. Los enlaces simbólicos son especialmente útiles cuando se quiere que un grupo de personas trabajen sobre un mismo fichero, puesto que permiten compartir el fichero pero centralizan las modificaciones.

2.6.3 El camino o path

En cualquier sistema operativo moderno la estructura de archivos es jerárquica y depende de los directorios. En general, la estructura del sistema de archivos se asemeja a una estructura de árbol, estando compuesto cada nudo por un directorio o carpeta, que contiene otros directorios o archivos.

En Windows, cada unidad de disco se identifica como una carpeta básica que sirve de raíz a otras y cuyo nombre es especial a:, C:, d: etc. En los sistemas UNIX y, por lo tanto, en Linux, existe una única raíz llamada */* de la que cuelgan todos los ficheros y directorios, y que es independiente de los dispositivos que estén conectados al ordenador.

El camino o *path* de un fichero o directorio es la secuencia de directorios que se ha de recorrer para acceder a un determinado fichero separados por */*. Supongamos la estructura de archivos de la siguiente figura:

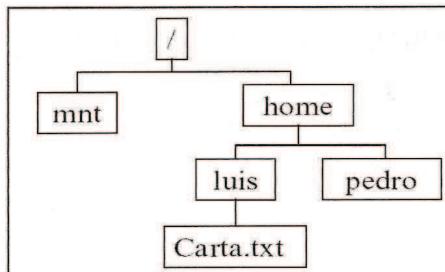


Figura. Árbol de Dirección.

Existen dos formas del path o camino:

- Camino absoluto que muestra toda la ruta a un fichero (por ejemplo, */home/luis/Carta.txt*).
- El *path* relativo a un determinado directorio (por ejemplo, si nos encontramos en el directorio */home*, el path relativo al fichero *Carta.lxt* es *luis/Carta.lxt*).

Además, todos los directorios contienen dos directorios especiales:

- El directorio actual, representado por el punto (.).
- El directorio padre representado por dos puntos (..).

Estando en el directorio */home/pedro* se puede acceder a *Carta.txt* con */home/luis/Carta.txt* (path absoluto) o bien *../luis/Carta.txt* (path relativo). En *luis* como *./Carta.txt* o simplemente *Carta.txt*.

2.6.4 Estructura del sistema de archivos de Linux

El sistema de archivos de Linux sigue todas las convenciones de UNIX lo que significa que tiene una estructura determinada, compatible y homogénea con el resto de los sistemas UNIX. Al contrario que en Windows o MS-DOS, el sistema de archivos en cualquier sistema UNIX no está ligado de una forma directa con la estructura del *hardware* (es decir, no depende de si un determinado ordenador tiene uno, dos o siete discos duros para crear las unidades *c:*, *d:* o *m:*).

Todos los sistemas de archivos de UNIX tienen un origen único la raíz o *root* representada por */*. Bajo este directorio se encuentran todos los ficheros a los que puede acceder el sistema operativo. Estos ficheros se organizan en distintos directorios cuya misión y nombre son estándar para todos los sistemas UNIX. Estos directorios son y se utilizan para:

- *1. Raíz del sistema de archivos.*

- **/dev.** Contiene ficheros del sistema representando los dispositivos que estén físicamente instalados en el ordenador.
- **/etc.** Este directorio está reservado para los ficheros de configuración de: sistema. En este directorio no debe aparecer ningún fichero binario (programas). Bajo este deben aparecer otros dos subdirectorios:
 - */etc/X11*. Ficheros de configuración de X Windows
 - */etc/skel*. Ficheros de configuración básica que son copiados al directorio del usuario cuando se crea uno nuevo.
- **/bin.** Es la abreviatura de *binaries* (ejecutables). Es donde residen *.L* mayoría de los programas esenciales del sistema.
- **/lib.** Contiene las imágenes de las librerías compartidas. Estos ficheros contienen código que compartirán muchos programas. En lugar de que cada programa contenga una copia propia de las rutinas compartidas, estas son guardadas en un lugar común, en */lib*. Esto hace que los programas ejecutables sean menores y se reduce el espacio usado en disco.
- **/proc.** Contiene ficheros especiales que reciben o envían información al *kernel* del sistema. Es un *sistema de ficheros virtual*. Los ficheros que contiene realmente residen en memoria, no en disco. Hacen referencia a varios procesos que corren en el sistema y le permiten obtener información acerca de qué programas y procesos están corriendo en un momento dado.
- **/var.** Este directorio contiene información temporal de los programas.
 - **/var/adrn.** Contiene varios ficheros de interés para el administrador del sistema, específicamente históricos del sistema, los cuales recogen errores o problemas con el sistema. Otros ficheros guardan las sesiones de presentación en el sistema, así como los intentos fallidos.
 - **/var/spool.** Contiene ficheros que van a ser pasados a otro programa. Por ejemplo, si la máquina está conectada a una red, el correo de llegada será almacenado en */var/spool/mail* hasta que se lea o se borre. Artículos nuevos de las news tanto salientes como entrantes pueden encontrarse en */var/spool/news*, etc.
- **/home.** Contiene los directorios *home* de los usuarios. Por ejemplo, */home/Alex* es el directorio del usuario Alex. En un sistema recién instalado, no habrá ningún usuario en este directorio.
- **/sbin.** Contiene programas que son únicamente accesibles al superusuario.

- **/usr**. Este es uno de los directorios más importantes del sistema puesto que contiene los programas de uso común para todos los usuarios. Su estructura suele ser similar a la siguiente.
 - **/usr/X11R6**. Contiene los programas para ejecutar X Windows
 - **/usr/bin**. Programas de uso general, como el compilador de C/C++.
 - **/usr/doc**. Documentación general del sistema.
 - **/usr/etc**. Contiene los diferentes ficheros de configuración y programas del sistema. En general, los ficheros que se encuentran en */usr/etc/* no son esenciales para el sistema, a diferencia de los que se encuentran en */etc*, que sí lo son.
- **/usr/include**. Contiene los ficheros de cabecera para el compilador de C. Estos ficheros (la mayoría de los cuales tienen la extensión .h, de *header*) declaran estructuras de datos, subrutinas y constantes usadas en la escritura de programas en C. Los ficheros que se encuentran en */usr/include/sys* son, generalmente, utilizados en la programación en UNIX a nivel de sistema.
- **/usr/info**. Ficheros de información de GNU.
- **/usr/lib**. Contiene las librerías equivalentes *stub* y *static* a los ficheros encontrados en */lib*. Al compilar un programa, este es «enlazado» con las librerías que se encuentran en */usr/lib*, las cuales dirigen al programa a buscar en */lib* cuando necesita el código de la librería. Además, varios programas guardan ficheros de configuración en */usr/lib*.
- **/usr/man**. Manuales accesibles con el comando man.
- **/usr/sbin**. Programas de administración del sistema.
- **/usr/src**. Código fuente de programas. En */usr/src/Linux*, está el código fuente del núcleo de Linux.
- **/usr/local**. Es muy parecido a */usr*, contiene programas y ficheros no esenciales para el sistema. En general, los programas que se encuentran en */usr/local* son específicos de su sistema (esto quiere decir que el directorio */usr/local* difiere bastante entre sistemas UNIX).

Existen además de los anteriores otros directorios que se suelen localizar: en el directorio */usr*, como por ejemplo las carpetas de los programas que se instalen en el sistema.

- **/tmp**. Muchos programas tienen la necesidad de generar cierta información

temporal y guardarla en un fichero temporal. El lugar habitual para esos ficheros es en `/tmp`.

2.7 Variables de entorno

Hay ocasiones en las que es útil almacenar ciertos valores cuando se trabaja en una sesión de Shell para que los puedan utilizar diferentes programas. Para realizar esta tarea, se usan las variables de entorno que son unas variables que existen en la sesión de *shell* abierta y que, por tanto, se perderán al cerrarla.

El método para asignar un valor a una variable es teclear en el *prompt* del shell su nombre seguido del signo igual y del valor que se quiere asignar.

```
$ nombre = valor
```

Para hacer referencia al valor de la variable se utiliza el símbolo `$` y el nombre de la variable. Una forma de comprobar su valor es mediante el comando echo:

```
$ echo $nombre
valor
```

Ejemplo

```
$
pi=3.14159

$ echo
$pi
3.14159
```

Para asignar a una variable una cadena que contenga espacios o caracteres especiales del sistema, hay que encerrarla entre comillas simples.

Ejemplo

```
$ dirección ='Calle San Agustín 11 numero 1'

$ echo $dirección
Calle San Agustín número 1
```

A las variables de entorno creadas por el usuario o por los programas para su uso propio, hay que añadir las variables intrínsecas del *shell* y que son necesarias para su buen funcionamiento. Una de las más utilizadas es la variable PATH que

contiene la ruta por defecto en la que se buscarán los comandos tecleados por el usuario. Las más usuales se pueden ver en la siguiente tabla:

Variable	Contenido
\$#	Número de argumentos recibidos
\$*	Todos los argumentos
\$-	Opciones dadas
\$?	Valor devuelto por el último comando ejecutado
\$\$	Identificador del proceso actual
\$HOME	Directorio del usuario
\$IFS	Lista de caracteres <i>Que</i> se utilizan como separadores en <i>los</i> argumentos
\$MAIL	Archivo donde se almacena el correo electrónico del usuario
\$PS1	Prompt del usuario
\$PS2	Prompt de continuación de línea
\$PATH	Lista de directorios de búsqueda para los comandos

Tabla. Variables de entorno más comunes

Ejemplo

```
$ echo $PATH
:/bin:/usr/bin:/home/alex/bin
```

El conocimiento y manejo de estas variables es fundamental en el trabajo desde el *prompt* de comandos del *shell* y, especialmente, al hacerlo con archivos de proceso por lotes.

2.8 Administración del sistema

Una vez instalado el sistema operativo, queda la labor que requiere una mayor dedicación: la correcta administración del mismo. En UNIX la persona encargada de administrar es el usuario *root*. Este usuario privilegiado utiliza una cuenta especial, la cuenta de *root*, que otorga todos los permisos posibles a quien la utiliza; desde esta cuenta es desde donde se han de realizar todas las funciones de administración de la máquina (creación de usuarios y grupos, instalación de periféricos, montado de sistemas de ficheros, etc.), ya que estas tareas requieren de permisos especiales.

Debido a sus características especiales esta cuenta sólo se ha de utilizar cuando sea estrictamente necesario, ya que cualquier error cometido puede dejar el sistema inutilizable. Así pues, el administrador debe tener otra cuenta con permisos estándar para el uso habitual del sistema y la cuenta de *root* únicamente para tareas administrativas.

2.8.1 Gestión de usuarios y grupos

Al ser UNIX un sistema multiusuario, debe proteger los ficheros de los usuarios particulares de la manipulación por parte de otros usuarios. Para esta labor, UNIX proporciona un mecanismo conocido como permisos de ficheros.

Por otra parte, todos los usuarios de UNIX deben tener una cuenta de usuario en el sistema que establezca los privilegios del mismo. A su vez, UNIX organiza a los usuarios en grupos, de forma que se puedan establecer privilegios a un determinado grupo de trabajo para el acceso a determinados archivos o servicios del sistema.

Será labor del administrador del sistema el diseñar las políticas oportunas para gestionar a los usuarios y definir los grupos a los que pueden pertenecer, asignándoles los privilegios y restricciones adecuados a la política del sistema.

El sistema mantiene cierta información acerca de cada usuario. Dicha información se resume a continuación:

- **Nombre de usuario.** El nombre de usuario es el identificador único dado a cada usuario del sistema. Ejemplos de nombres de usuario son alex, pepe y amacc. Se pueden utilizar letras y dígitos junto a los caracteres «-» (subrayado) y «.» (punto). Los nombres de usuario se limitan normalmente a 8 caracteres de longitud.
- **user ID.** El user ID (o UID) es un número único dado a cada usuario del sistema. El sistema identifica al propietario de los procesos por UID, no por nombre de usuario.
- **group ID.** El group ID (o GID) es la identificación del grupo del usuario por defecto. Cada usuario pertenece a uno o más grupos definidos por el administrador del sistema.
- **Clave.** El sistema también almacena la clave encriptada del usuario.
- **Nombre completo.** El nombre real del usuario se almacena junto con el nombre de usuario.
- **Directorio inicial.** El directorio inicial es el directorio en el que se coloca inicialmente al usuario en tiempo de conexión. Cada usuario debe tener su propio directorio inicial, normalmente situado bajo `/home` y cuyo valor se almacena en la variable de entorno `$HOME`.
- **Intérprete de inicio.** El intérprete de inicio del usuario es el intérprete de

comandos que es ejecutado para el usuario en tiempo de conexión.

El fichero `/etc/passwd` es el fichero en el que se almacena la información anterior acerca de los usuarios. Cada línea del fichero contiene información acerca de un único usuario; el formato de cada línea es:

nombre:clave encriptada:UID:GID: nombre e completo:dir. inicio: intérprete

Un ejemplo puede ser:

jcalvo:E45Q981 g71 oAds: 106:100:Juan Calvo:/home/jcalvo:/bin/bash

El primer campo (*jcalvo*) es el nombre de usuario.

El segundo campo (*E45Q981g71oAds*) es la clave encriptada. Las claves no se almacenan en el sistema en ningún formato legible. Las claves se encriptan utilizándose a sí mismas como clave secreta. En otras palabras, sólo si se conoce la clave, ésta puede ser desencriptada. Esta forma de encriptación se denomina encriptación no reversible y es bastante segura. Algunos sistemas utilizan *claves en sombra* en la que la información de las claves se relega al fichero `/etc/shadow`. Puesto que `/etc/passwd` es legible por todo el mundo, `/etc/shadow` suministra un grado extra de seguridad, puesto que este fichero no lo es. Las claves en sombra suministran algunas otras funciones como puede ser la expiración de claves.

El tercer campo (*106*) es el **VID**. Este debe ser único para cada usuario.

El cuarto campo (*100*) es el **GID**. Este usuario pertenece al grupo numerado 100. La información de grupos, como la información de usuarios, se almacena en el fichero `/etc/group`.

El quinto campo es el nombre completo del usuario (*Juan Calvo*). Los dos últimos campos son el directorio inicial del usuario (`/home/jcalvo`) y el intérprete de conexión (`/bin/bash`), respectivamente. No es necesario que el directorio inicial de un usuario tenga el mismo nombre que el del nombre de usuario. Sin embargo, ayuda a identificar el directorio.

Para añadir un usuario al sistema hay que seguir varios pasos. Primero, se debe crear una entrada en `/etc/passwd`, con un nombre de usuario y UID únicos. Se debe especificar el GID, nombre completo y resto de información. Se debe crear el directorio inicial y poner los permisos en el directorio para que el usuario sea el dueño. Se deben suministrar los ficheros de comandos de inicialización en el nuevo directorio y se debe hacer alguna otra configuración del sistema (por ejemplo, preparar un buzón para el correo electrónico entrante para el nuevo usuario).

Aunque no es difícil añadir usuarios a mano, cuando se está ejecutando un sistema con muchos usuarios, es fácil olvidarse de algo. La manera más simple de añadir usuarios es utilizar un programa interactivo que vaya preguntando por la información necesaria y actualice todos los ficheros del sistema automáticamente. El nombre de este programa es **useradd** o **adduser** dependiendo del software que esté instalado. Las páginas **man** (ayuda) para estos comandos son suficientemente autoexplicatorias. También existen asistentes gráficos en cada sistema de ventanas que facilitan enormemente la labor.

De forma análoga, borrar usuarios puede hacerse con los comandos **userdel** o **deluser** dependiendo del *software* que hubiera instalado en el sistema.

Una acción que se ha de realizar en varias ocasiones es la de cancelar momentáneamente una cuenta. Si se desea deshabilitar temporalmente a un usuario para evitar que se conecte al sistema (sin necesidad de borrar su cuenta del usuario), se puede prefijar con un asterisco (*) el campo de la clave en */etc/passwd*, de tal forma que no coincida con la original al realizar el desencriptado.

Por ejemplo, cambiando la línea de */etc/passwd* correspondiente *ajcalvo* a:

*jcalvo: *E45Q981g71 oAds: 1 06: 100:Juan Calvo:/home/jcalvo:/bin/bash*

Cuando se necesite modificar alguno de los datos de usuario se puede realizar directamente editando el fichero */etc/passwd* o mediante alguno de los programas o comandos antes mencionados exceptuando la contraseña o *password* que habrá que cambiar usando el comando *passwd*, ya que se ha de encriptar usando el algoritmo de encriptación.

Los usuarios pueden cambiar su propia clave, pero sólo el usuario *root* puede cambiar la clave de otro usuario.

En algunos sistemas, los comandos **chfn** y **chsh** están disponibles y permiten a usuarios cambiar sus atributos de nombre completo e intérprete de conexión. Si no, deberán pedir al administrador de sistemas que se los cambie.

Como se ha citado anteriormente, cada usuario pertenece a uno o más grupos. La única importancia real de las relaciones de grupo es la perteneciente a los permisos de ficheros. Como se verá en la siguiente sección, cada fichero tiene un *grupo propietario* y un conjunto de permisos de grupo que define de qué forma pueden acceder al fichero los usuarios del grupo.

Hay varios grupos definidos en el sistema, como pueden ser *bin*, *mail* y *sys*. Los usuarios no deben pertenecer a ninguno de estos grupos; se utilizan únicamente

para permisos de ficheros del sistema. En su lugar, los usuarios deben pertenecer a un grupo individual, como *users*.

El fichero */etc/group* contiene información acerca de los grupos. El formato de cada línea es:

nombre de grupo:clave:GID:otros miembros

Algunos ejemplos de grupos son:

```
root: *:0:
usuarios:*:100:jcalvo, alex
invitados:*:200:
otros:*:250:virtu
```

El primer grupo (*root*) es un grupo especial del sistema reservado para la cuenta root. El siguiente grupo (*users*) es para usuarios normales. Tiene un GID de 100. Los usuarios *jcalvo* y *alex* tienen acceso a este grupo. Recuérdese que en */etc/passwd* cada usuario tiene un GID por defecto. Sin embargo, los usuarios pueden pertenecer a más de un grupo, añadiendo sus nombres de usuario a otras líneas de grupo en */etc/group*. El comando **groups** lista a qué grupos pertenece cada usuario.

El tercer grupo (*invitados*) es para usuarios invitados y *otros* es para otros usuarios. El usuario *virtu* tiene acceso a este grupo.

Como se puede ver, el campo *clave* de */etc/group* raramente se utiliza. A veces, se usa para dar una clave para acceder a un grupo. Esto es raras veces necesario. *Para evitar* que los *usuarios* cambien a grupos *privilegiados* (con el comando **newgroup**), se pone el campo de la clave a *.

Se pueden usar los comandos **addgroup** o **groupadd** para añadir grupos al sistema. Normalmente, es más sencillo añadir líneas a */etc/group* directamente, puesto que no se necesitan más configuraciones para añadir un grupo. Para borrar un grupo, sólo hay que borrar su entrada de */etc/group*.

2.8.2 Permisos

Linux, al igual que todos los sistemas UNIX, mantiene un sistema de permisos de acceso a los ficheros muy estricto, con el fin de controlar qué es lo que se puede hacer con ellos y quién lo puede hacer. Estos permisos se identifican con letras y son:

r permiso de lectura en el fichero

- w permiso de escritura en el fichero
- x permiso de ejecución del fichero
- s permiso para cambiar el propietario del fichero

Al contrario que en Windows o MS-DOS, los programas ejecutables de Linux no están marcados por una determinada extensión (.exe, o .com) sino por un atributo, el permiso de ejecución x. Si se elimina este atributo a un programa, Linux no será capaz de ejecutado. A su vez, cada uno de estos permisos se aplica: al dueño de fichero (*u*), al grupo de usuarios al que pertenece el dueño (*g*) o al resto de usuarios (*a*). Así, un fichero determinado puede tener permiso para ser leído, escrito: ejecutado por su dueño, leído y ejecutado por el grupo al que pertenece y no tener ningún tipo de acceso para los demás usuarios. Como se puede entender, este tipo de mecanismo es especialmente útil cuando se trabaja en grupo en un determinado proyecto.

Este mecanismo permite que ficheros y directorios pertenezcan a un usuario en particular. Por ejemplo, si el usuario Alex creó ficheros en su directorio *home*, Alex es el propietario de esos ficheros y tiene acceso a ellos.

UNIX también permite que los ficheros sean compartidos entre usuarios y grupos de usuarios.

Si Alex lo desea, podría restringir el acceso a sus ficheros de forma que ningún otro usuario tenga acceso. De cualquier modo, en la mayoría de los sistemas, por defecto se permite que otros usuarios puedan leer los ficheros de otros usuario, pero no modificados o borrarlos.

Como se ha explicado arriba, cada fichero pertenece a un usuario en particular. Por otra parte, los ficheros también pertenecen a un grupo en particular, que es un conjunto de usuarios definido por el sistema. Cada usuario pertenece al menos a un grupo cuando es creado. El administrador del sistema puede hacer que un usuario tenga acceso a más de un grupo.

Los grupos usualmente son definidos por el tipo de usuarios que acceden a la máquina. Por ejemplo, en un sistema UNIX de una universidad, los usuarios pueden ser divididos en los grupos estudiantes, dirección, profesores e invitados. Hay también unos pocos grupos definidos por el sistema (como *bin* y *admin*) los cuales son usados por el propio sistema para controlar el acceso a los recursos. Muy raramente los usuarios normales pertenecen a estos grupos.

Los permisos están divididos en tres tipos: lectura, escritura y ejecución. Estos permisos pueden ser fijados para tres clases de usuarios: el propietario del fichero (user o usuario), el grupo al que pertenece el propietario del fichero

(group o grupo) y para todos los usuarios del sistema independientemente del grupo.

El permiso de lectura permite a un usuario leer el contenido del fichero o, en el caso de un directorio, listar el contenido del mismo (usando el comando `/s`). El permiso de escritura permite a un usuario escribir y modificar el "fichero. Para directorios, el permiso de escritura permite crear nuevos ficheros o borrar ficheros ya existentes en dicho directorio. Por último, el permiso de ejecución permite a un usuario ejecutar el fichero si es un programa o script (guión) del intérprete de comandos. Para directorios, el permiso de ejecución permite al usuario cambiar al directorio en cuestión (con el comando `cd`).

Se presenta un ejemplo clarificador del uso de permisos de ficheros.

EJEMPLO

Usando el comando `ls` con la opción `-l` se mostrará un listado largo de los ficheros, el cual incluye los permisos de ficheros.

```
# /s -l stuff
-rw-r--r-- 1 Alex users 505 Mar 13 19:05 stuff
```

El primer campo impreso en el listado representa los permisos del fichero, el tercer campo indica quién es el propietario del fichero (Alex), y el cuarto es el grupo al cual pertenece el fichero (*users*). Obviamente, el último campo es el nombre del fichero (*stuff*). Los demás campos no son relevantes en este momento.

Este fichero pertenece a Alex y al grupo *users* y la cadena `-rw-r--r--` informa, por orden, de los permisos para el propietario, el grupo del fichero y cualquier otro usuario.

El primer carácter de la cadena de permisos (-) representa el tipo de fichero. Significa que es un fichero regular; otras opciones para este campo son *d* que indica que se trata de un directorio, / que indica que se trata de un archivo de enlace (*link* en inglés) y *s* que indica que se trata de un enlace simbólico. Las siguientes tres letras (*rw-*) representan los permisos para el propietario del fichero, Alex en este ejemplo. El *r* para lectura y *w* para escritura. Luego Alex tiene permisos de lectura y escritura para el fichero *stuff*.

Como ya se ha mencionado, aparte de los permisos de lectura y escritura está el permiso de ejecución, representado por una *x*. Como hay un *-* en lugar del *x*, significa que Alex no tiene permiso para ejecutar ese fichero. *Esto es correcto, puesto que stuff*.

Como ya se mencionado, a parte de los permisos de lectura y escritura, está el permiso de ejecución, representado por una *x*. Como hay un *-* en lugar del *x*, significa que Alex no tiene permiso para ejecutar ese fichero. *Esto es correcto, puesto que stuff no es un programa ejecutable.* Por supuesto, como Alex es el propietario del fichero, puede darse permiso de ejecución si lo desea.

Los siguientes tres caracteres, (*r--*) representan los permisos para los miembros del grupo. El grupo al que pertenece el propietario del fichero y por ende el fichero es *users*. Como sólo aparece un *r* cualquier usuario que pertenezca al grupo *users* puede leer este fichero.

Análogamente, los últimos tres caracteres, también (*r--*), representan los permisos para cualquier

otro usuario del sistema (diferentes del propietario o de los pertenecientes al grupo *users*). De nuevo, como sólo está presente el *r*, los demás usuarios pueden leer el fichero, pero no escribir en él o ejecutarlo.

Aquí tenemos otros ejemplos de permisos de grupo:

-rwxr-xr-x El propietario del fichero puede leer, escribir y ejecutar el fichero. Los usuarios pertenecientes al grupo del fichero, y todos los demás usuarios pueden leer y ejecutar el fichero.

-rw----- El propietario del fichero puede leer y escribir. Nadie más puede acceder al fichero.

-rwxrwxrwx Todos los usuarios pueden leer, escribir y ejecutar el fichero

Es importante darse cuenta de que los permisos de un fichero también dependen de los permisos del directorio en el que residen. Por ejemplo, aunque un fichero tenga los permisos *-rwxrwxrwx*, otros usuarios no podrán acceder a él a menos que también tengan permiso de lectura y ejecución para el directorio en el cual se encuentra el fichero. Si Alex quiere restringir el acceso a todos sus ficheros, podría simplemente poner los permisos de su directorio *home /home/Alex* a *drwx-----*. De esta forma, ningún usuario podrá acceder a su directorio ni a ninguno de sus ficheros o subdirectorios. Esto se conoce con el nombre de dependencia.

En otras palabras, para acceder a un fichero se debe tener permiso de ejecución de todos los directorios a lo largo del camino de acceso al fichero, además de permiso de lectura (o ejecución) del fichero en particular.

Habitualmente, los usuarios de un sistema UNIX son muy abiertos con sus ficheros. Los permisos que se dan a los ficheros usualmente son *-rw-r--r-*, permite a todos los demás usuarios leer los ficheros, pero no modificarlos de ninguna forma. Los directorios usualmente tienen los permisos *drwxr-xr-x*, lo que permite que los demás usuarios puedan moverse y ver los directorios, pero sin poder crear nuevos ficheros en ellos.

Muchos usuarios pueden querer limitar el acceso de otros usuarios a sus ficheros. Poniendo los permisos de un fichero a *-rw-----* no se permitirá a ningún otro usuario (salvo al usuario root) acceder al fichero. Igualmente, poniendo los mismos del directorio a *drwx-----* no se permitirá a los demás usuarios acceder al directorio en cuestión.

2.8.3 Actualizando e instalando nuevo *software*

Otra tarea del administrador del sistema es la actualización e instalación de nuevo *software*.

La comunidad Linux es muy dinámica. Las versiones nuevas del núcleo aparecen cada pocos meses y otros programas se actualizan casi tan a menudo. Pero no es imprescindible actualizar sólo porque exista una versión más moderna. Una actualización supone mucho esfuerzo en tiempo y recursos, por lo que sólo se ha de realizar si las mejoras aportadas por la nueva versión son significativas y necesarias para nuestro sistema.

La mejor forma de actualizar su sistema es haciéndolo a mano: actualizando sólo aquellos paquetes de *software* que haya que actualizar.

El *software* más importante para actualizar en el sistema es el núcleo, las librerías y el compilador *gcc*. Estas son las tres partes esenciales del sistema y en algunos casos cada uno depende de las otras para que todo funcione bien. La mayor parte del resto del *software* del sistema no necesita ser actualizado a menudo.

Cada paquete de *software* tiene su propio sistema de instalación, por lo que es imposible citar aquí la forma de instalados todos; como norma se debe examinar cuidadosamente la documentación incluida al respecto en el nuevo *software* y seguir escrupulosamente los pasos allí marcados. También, es posible ayudarse de sistemas de instalación incluidos en algunas de las distribuciones como los ya comentados *RPM* o *DEB*. No hace falta indicar que para la instalación de *software* crítico se han de tener los permisos de *root*.

2.8.4 Instalación de dispositivos

Además del *software*, en el sistema se han de instalar diferentes dispositivos hardware. Como ya se ha explicado anteriormente, UNIX trata todos y cada uno de los dispositivos conectados al sistema como si de archivos se tratara, lo que hace que su uso sea igual independientemente del dispositivo. En lo que respecta a la instalación, también sigue un proceso análogo y relativamente sencillo que consiste en editar un fichero determinado, que depende del tipo de *hardware* a instalar, sito en el directorio */etc* en el que se indican las características del dispositivo y el archivo del sistema que referencia al dispositivo, generalmente ubicado en */dev*.

Para ilustrar este método, se expone en detalle la instalación de dos de los tipos de dispositivos más comunes, las impresoras y los sistemas de almacenamiento, como pueden ser disquetes, discos duros, unidades de CD-ROM, etc.

2.8.5 Acceso a los diferentes sistemas de almacenamiento

Como se ha visto anteriormente, el sistema de archivos de Linux sólo tiene una raíz y su estructura es independiente de los dispositivos de almacenamiento existentes. Esto implica que el procedimiento a emplear para acceder a la información almacenada en los distintos sistemas de almacenamiento de un ordenador no es tan sencillo como en Windows y requiere de un proceso llamado *montado*. Cuando se ha terminado de trabajar con un determinado dispositivo hay que invertir el proceso y *desmontar/o*.

Por ejemplo, el proceso para leer un disquete sería el siguiente:

1. Introducir el disquete en la disquetera.
2. Montar el sistema de archivos del mismo.
3. Leer, grabar, o manipular el contenido del disquete.
4. Desmontar el sistema de archivos del disquete.
5. Extraer el disquete de la disquetera.

El proceso puede parecer complejo pero es el precio a pagar por la seguridad, puesto que de esta forma se garantiza que no exista ninguna aplicación que esté usando el disquete cuando se extraiga.

Sólo el administrador o root, por motivos de seguridad, tiene permiso para montar y desmontar un sistema de archivos.

2.9 Montando sistemas de ficheros

Antes de que un sistema de ficheros sea accesible al sistema, debe ser montado en algún directorio. Por ejemplo, si se tiene un sistema de ficheros en un disquete, se debe montar bajo algún directorio, generalmente */mnt*, para poder acceder a los ficheros que contiene. Después de montar el sistema de ficheros, todos los ficheros en dicho sistema aparecen en ese directorio. Tras desmontar el sistema de ficheros, el directorio (en este caso, */mnt*) estará vacío.

Lo mismo es válido para los sistemas de ficheros del disco duro. El sistema monta automáticamente los sistemas de ficheros del disco duro en tiempo de arranque. El así llamado «sistema de ficheros raíz» es montado en el directorio */*. Si se tiene un sistema de ficheros separado para */usr*, por ejemplo, se monta en */usr*. Si sólo se tiene un sistema de ficheros raíz, todos los ficheros (incluyendo los de */usr*) existen en ese sistema de ficheros.

El comando **mount** se utiliza para montar un sistema de ficheros. El comando **mount -av** se ejecuta desde el fichero */etc/rc* (que es el fichero de inicialización del sistema, ejecutado en tiempo de arranque). El comando **mount -av** obtiene información de los sistemas de ficheros y puntos de montaje del fichero */etc/fstab*.

Este es un ejemplo de fichero *fstab*:

```
# dispositivo directorio tipo opciones
/dev/hda2 / ext2 defaults
/dev/hda3 /usr ext2 defaults
/dev/hda4 none swap sw
/proc /proc proc none
```

El primer campo es el dispositivo, el nombre de la partición a montar. El segundo campo es el punto de montaje. El tercero es el tipo de sistema de ficheros como puede ser *ext2* (para *ext2ft*). La tabla lista algunos de los tipos de sistemas de ficheros de Linux.

Sistema de ficheros	Tipo	Comentarios
Second Extended Filesystem	ext2	Sistema de ficheros mas común en Linux.
Extended Filesystem	ext	Reemplazado por ext2.
Minix Filesystem	mmiX	Sistema de ficheros Minix original; raras veces utilizado.
Xia Filesvstem	Xla	Como ext2, pero raras veces utilizado.
UMSDOS Filesystem	umsdos	Utilizado para instalar Linux en una partición MS-DOS.
MS-DOS Filesystem	msdos	Utilizado para acceder a ficheros MS-DOS.
/proc Filesystem	proc	Suministra información de proceso para ps, etc.
ISO 9660 Filesystem	iso9660	Frecuentemente utilizado por muchos CD-ROMs.
Xenix Filesystem	xenix	Sistema de ficheros de Xenix.
System V Filesystem	svsv	Variantes del System V para el x86.
Coherent Filesystem	coherent	Acceso a ficheros de Coherent.
HPFS Filesystem	hpfs	Acceso en lectura a particiones HPFS.

Tipos de sistemas de ficheros Linux

El último campo del fichero *fstab* contiene las opciones del comando *mount* normalmente, está puesto a *defaults* (por defecto).

Como se puede ver, las particiones de intercambio están incluidas en */etc/fstab* también. Tienen un punto de montaje *none* y tipo *swap*. El comando *swapon -a*, que se ejecuta también desde */etc/re*, se utiliza para activar el intercambio en todos los dispositivos de intercambio listados en */etc/fstab*.

El fichero *fstab* contiene una entrada especial para el sistema de ficheros */proc*, el sistema de ficheros */proc* se utiliza para almacenar información acerca de los procesos del sistema, memoria disponible y otros datos del mismo tipo. Si */proc* no está montado, no funcionarán comandos como *ps* (comando que proporciona información de los procesos activos en el sistema).

El comando *mount* sólo puede ser utilizado por root. Esto es así para garantizar la seguridad del sistema. Aunque y para facilitar la usabilidad de los equipos, existen varios paquetes que permiten a los usuarios normales montar y desmontar sistemas de ficheros (disquetes en particular) sin comprometer la seguridad del sistema.

El comando *mount -av* realmente monta todos los sistemas de ficheros excepto el sistema de ficheros raíz (en el ejemplo anterior, */dev/hda2*). El sistema de ficheros raíz es montado automáticamente en tiempo de arranque por el núcleo.

En vez de utilizar el comando *mount -av*, se puede montar un sistema de ficheros a mano. El comando *# mount -t ext2 /dev/hda3 /usr* es equivalente a montar el sistema de ficheros con la entrada */dev/hda3* del ejemplo de fichero *fstab* anterior.

Los sistemas de ficheros son desmontados bien manualmente mediante el comando **umount** o automáticamente por los comandos **shutdown** o **halt** antes de cerrar el sistema.

Normalmente, es una buena idea el comprobar de vez en cuando los sistemas de ficheros en busca de ficheros dañados o corruptos. Algunos sistemas comprueban automáticamente sus sistemas de ficheros en tiempo de arranque (con los comandos apropiados en */etc/rc*).

Normalmente, es una buena idea el comprobar de vez en cuando los sistemas de ficheros en busca de ficheros dañados o corruptos. Algunos sistemas comprueban automáticamente sus sistemas de ficheros en tiempo de arranque (con los comandos apropiados en */etc/rc*).

```
# e2fsck -av /dev/hda2
```

comprobará el sistema de ficheros *ext2fs* de */dev/hda2* y corregirá automáticamente cualquier error.

Habitualmente, es una buena idea desmontar un sistema de ficheros antes de - comprobarlo. Por ejemplo, el comando:

```
# umount /dev/hda2
```

desmontará el sistema de ficheros en */dev/hda2*, tras lo cual podrá ser comprobado. La única excepción es que no se puede desmontar el sistema de ficheros raíz. Para poder comprobar el sistema de ficheros raíz cuando está desmontado, se debe utilizar un disquete de arranque/raíz. Tampoco se puede desmontar un sistema de ficheros si alguno de sus ficheros está siendo utilizado por un proceso en ejecución.

Por ejemplo, no se puede desmontar un sistema de ficheros si el directorio de

trabajo de algún usuario está en ese sistema de ficheros. Se recibirá un error *Device busy* (dispositivo ocupado) si se intenta desmontar un sistema de ficheros que esté en uso.

Otros tipos de sistemas de ficheros utilizan sus propios comandos, como puede ser **efsck** y **xfsck**. En algunos sistemas, se puede utilizar el comando **fsck**, que determina el tipo de sistema de ficheros y ejecuta el comando apropiado.

Es importante reiniciar el sistema inmediatamente después de comprobar un sistema de ficheros montado, si es que se hizo alguna corrección al sistema de ficheros (aunque, como ya se ha dicho, es preferible comprobar sistemas de ficheros que no estén montados), esto permite al sistema resincronizar su información acerca del sistema de ficheros cuando *e2fsck* lo modifica.

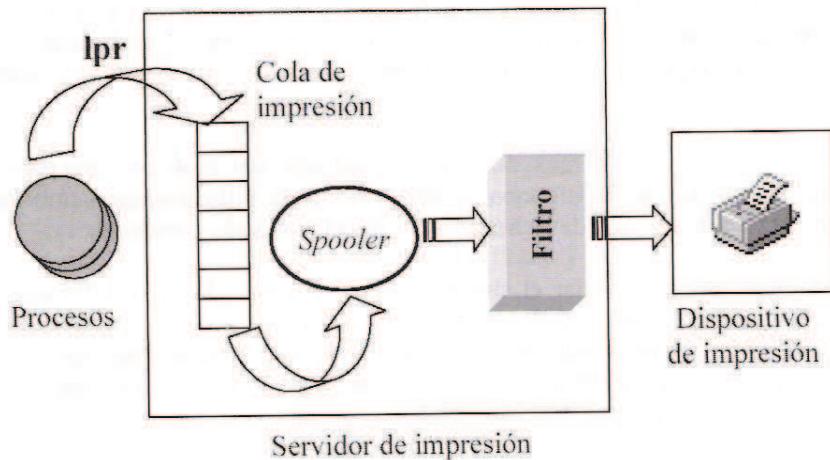
El sistema de ficheros */proc* no necesita nunca ser comprobado de esta forma, ya que */proc* es un sistema de ficheros en memoria, gestionado directamente por el núcleo.

2.10 Instalación de impresoras y gestión de la impresión

Otra de las labores del administrador es la instalación de las impresoras y la gestión de la impresión. En UNIX y Linux la gestión de la impresión se realiza a través de colas de impresión. En cada sistema existe un proceso ejecutándose constantemente que es el encargado de gestionar estas colas de impresión del sistema. Este demonio (*daemon* en inglés), llamado spooler, se encarga entre otras cosas de secuenciar las tareas de impresión y de convertir los datos a un formato o lenguaje de impresión compatible con la impresora.

Entre estos lenguajes de impresión, los más habituales son:

- Texto ASCII. Universal para todas las impresoras.
- PostScript. Lenguaje de impresión estándar de UNIX.
- Utilización de filtros (*GhostScript*) para convertir PostScript al lenguaje de impresión. El *GhostScript* se utiliza para imprimir documentos PostScript cuando no se dispone de una impresora compatible PostScript, dispone de una multitud de drivers para soportar amplia variedad de impresoras. Permite preparar documentos para su visualización en pantalla.
- PCL5 y PCL6 para impresoras HP.
- ESC/P, ESC/P2 y ESC/P Raster para impresoras Epson.



Las colas de impresión se definen en el archivo `/etc/printcap`. En este archivo se especifica el archivo de dispositivo asociado a cada impresora y el filtro que se asocia a la cola de impresión. También se definen otros parámetros como el tipo de papel que se utiliza o la resolución de la impresora. Por último, se asigna un tamaño para la cola de cada impresora.

Es posible definir varias colas de impresión, tanto si se dispone de una única impresora como de varias. En este último caso es obvio, ya que se debe crear al menos una cola por impresora, pero es interesante poder crear varias colas por impresora para así poder definir distintos filtros y atributos para una impresora.

Ejemplo de archivo `/etc/printcap`:

```
ascii/lp1/PS_600dpi-a4-ascii-mono-600/PS_600dpi a4 ascii mono 600:
:lp=/dev/lp0:
:sd=/var/spool/lpd/PS_600dpi-a4-ascii-mono-600:
:lf=/var/spool/lpd/PS_600dpi-a4-ascii-mono-600/1og:
:af=/var/spool/lpd/PS_600dpi-a4-ascii-mono-600/acct:
:if=/var/lib/apsfilter/bin/PS_600dpi-a4-ascii-mono-600:
:la@:mx#0:
:tr=:cl:sh:sf:
lp/lp2/PS_600dpi-a4-auto-mono-600/PS_600dpi a4 auto mono 600:
:lp=/dev/lp0:
:sd=/var/spool/lpd/PS_600dpi-a4-auto-mono-600:
:lf=/var/spool/lpd/PS_600dpi-a4-auto-mono-600/1og:
:af=/var/spool/lpd/PS_600dpi-a4-auto-mono-600/acct:
:if=/var/lib/apsfilter/bin/PS_600dpi-a4-auto-mono-600:
:la@:mx#0:
:tr=:cl:sh:sf:

raw/lp2/PS_600dpi-a4-auto-mono-600dpi a4 raw mono 600:
:sd=/var/spool/lpd/PS_600dpi-a4-raw:
:lf=/var/spool/lpd/PS_600dpi-a4-raw/1og:
```

```
:af=/var/spool/lpd/PS_600dpi-a4-raw/acct:  
:if=/var/lib/apsfilter/bin/PS 600dpi-raw:  
:la@:mx#0:  
:tr=:cl:sh:sf:
```

Como puede verse en el ejemplo, una impresora está configurada usando 3 tipos diferentes de filtros:

- **ascii**, para ser usada con el filtro ASCII.
- **lp**, para ser usada con el filtro más adecuado a la impresora (el filtro se elige automáticamente).
- **raw**, para ser usada sin conversión de los datos (llamado en castellano modo *crudo*).

También se indica que la resolución de impresión es de 600 puntos por pulgada (600 dpi) Y el tamaño de papel será A4.

La conexión física de las impresoras se puede realizar mediante diferentes puertos:

- **Puerto Paralelo.** El dispositivo se vincula a un archivo en el directorio /dev con el nombre lp0, lp 1, lp

Si la conexión está bien realizada, se debe poder imprimir mediante este comando:

Echo -en "Hola">> dev/lp0

- **Puerto serie.** Se vincula a un archivo en /dev con el nombre ser0, ser1, ser2...
- **Puerto USB.** Se vincula a un archivo en /dev con el nombre usblp0, usblp1, usblp2...

La prueba de impresión se realiza con el comando:

Echo -en "Hola">> dev/usblp0.

También se puede gestionar una impresora conectada remotamente a través de los protocolos TCPIIP, para ello se ha de definir en el cliente (máquina local) una cola para gestionar la impresora. Esto se realiza también en el fichero /etc/printcap. Ejemplo:

```
nombre-Imp!remote printer on maquinaremota.dominio:  
:sd=/var/spool/lpd/nombre-Imp:  
:rm=maquinaremota.dominio:
```

```
:rp=remoto:  
:bk:sh:mk#0:
```

En el servidor de impresión hay que habilitar explícitamente a las máquinas que tienen permiso para imprimir dando de alta sus IPs en el archivo */etc/hosts.lpd*.

Para completar el proceso de impresión, el sistema dispone de otro demonio llamado *lpd* (*line printer daemon*) que se activa durante el arranque y que se está ejecutando constantemente en segundo plano. Este proceso inspecciona el archivo */etc/printcap*, es decir las colas de impresión definidas en el sistema y realiza las siguientes tareas:

- Organización de las colas locales (filtros y puertos de impresión).
- Ordenación de los trabajos en la cola de impresión.
- Supervisión del estado de las colas.
- Traspaso de solicitudes de colas remotas al *lpd* remoto.
- Recogida de solicitudes de máquinas remotas y dirigidas a colas locales.

El usuario, por su parte, dispone de ciertas utilidades para interactuar con el sistema de impresión.

- Para lanzar los trabajos de impresión, dispone del comando *lpr*.

Ejemplo:

```
$ lpr -Pcola archivo
```

1. *lpr* almacena los datos a imprimir en la cola de impresión.
2. El spooler pasa de la cola al filtro de impresión.
3. Se envían los datos una vez convertidos a la impresora.
4. Al finalizar, se elimina el trabajo de la cola de impresión.

- Para listar los trabajos existentes en una cola de impresión se puede usar el comando */pq*

```
$ /pq -Pcola
```

- Para eliminar un trabajo de una cola de impresión se puede usar el comando */prm*

```
$ /prm -Pco/a 234
```

donde *-Pco/a* es el nombre de la cola y 234 es el número del trabajo en la cola.

Aunque en este capítulo se ha explicado siempre el método «manual» para instalar dispositivos, ya que es, además del más estándar, el más didáctico en cuanto permite hacerse una idea más aproximada de cómo funciona el sistema, actualmente existen múltiples aplicaciones gráficas que asisten en esta tarea insertando ellas mismas las líneas necesarias en los archivos de configuración y que permiten tanto un uso como una gestión gráfica y más intuitiva de los dispositivos.

2.11 Resumen de órdenes básicas

Esta sección introduce algunas de las órdenes básicas más útiles de un sistema UNIX.

Nótese que las opciones usualmente comienzan con "-" y, en la mayoría de los casos, se pueden añadir múltiples opciones de una letra con un único "-". Por ejemplo, en lugar de usar *ls -l -F* es posible usar */s -F*.

En lugar de listar todas las opciones disponibles para cada uno de los comandos sólo se hablará de aquellas más útiles o importantes. De hecho, la mayoría de las órdenes tienen un gran número de opciones aunque sólo se usan un puñado de ellas. Se puede usar *man* para ver las páginas de manual de cada orden, donde se mostrará la lista completa de opciones disponibles.

Nótese también que la mayoría de las órdenes toman una lista de ficheros o directorios como argumentos, indicados como "*<fichero> . . <.fichero N>*". Por ejemplo, la orden *cp* toma como argumentos la lista de ficheros a copiar, seguidos del fichero o directorio destino. Cuando se copia más de un fichero, el destino debe de ser *un directorio*.

- **cd.** Cambia el directorio de trabajo actual

Sintaxis: *cd <directorio>*

<directorio> es el directorio al que cambiamos (se refiere al directorio actual y al directorio padre).

Ejemplo: *cd ..casa* pone *..casa* como directorio actual.

- **ls.** Muestra información sobre los ficheros o directorios indicados.

Sintaxis: *ls <fichero> <fichero2> . . <ficheroN>*

Donde *<fichero>* a *<ficheroN>* son los ficheros o directorios a listar.

Opciones: hay multitud de ellas.

Las más usadas comúnmente son: *-F* (usada para mostrar información sobre el tipo de fichero), y *-l* (da un listado largo incluyendo tamaño, propietario, permisos, etc.).

Ejemplo: *ls -l /home/jcalvo* mostrará el contenido del directorio */home/jcalvo*.

- **cp.** Copia fichero(s) en otro fichero o directorio.

Sintaxis: *cp <fichero> <fichero2> ... <ficheroN> <destino>*

Donde *<fichero>* a *<ficheroN>* son los ficheros a copiar, y *<destino>* es el fichero o directorio destino.

Ejemplo: *cp ./docs javi* copia el fichero *./docs* al fichero o directorio *javi*.

- **mv.** Mueve fichero(s) a otro fichero o directorio. Es equivalente a una copia seguida del borrado del original. Puede ser usado para renombrar ficheros, como el comando MS-DOS *RENAME*

Sintaxis: *mv <fichero> <fichero2> ... <ficheroN> <destino>*

Donde *<fichero1>* a *<ficheroN>* son los ficheros a mover y *<destino>* es el fichero o directorio destino.

Ejemplo: *mv ./docs javi* mueve el fichero *./docs* al fichero o directorio *javi*.

- **rm.** Borra ficheros. Nótese que cuando los ficheros son borrados en UNIX, son irrecuperables (a diferencia de MS-DOS, donde usualmente se puede recuperar un fichero borrado).

Sintaxis: *rm <fichero> <fichero2> ... <ficheroN>*.

Donde *<fichero1>* a *<ficheroN>* son los nombres de los ficheros a borrar.

Opciones: *-i* pedirá confirmación antes de borrar un fichero.

Ejemplo: *rm -i /home/jcalvo/javi /home/jcalvo/docs* borra los *ficheros javi* y *docs* en */home/jcalvo*.

- **mkdir.** Crea directorios nuevos.

Sintaxis: *mkdir <dir1> <dir2> . . . <dirN>*

Donde *<dir1>* a *<dirN>* son los directorios a crear.

Ejemplo: *mkdir /home/jcalvo/test* crea el directorio *test* colgando de */home/jcalvo*.

- **rmdir.** Esta orden borra directorios vacíos. Al utilizar *rmdir*, el directorio de trabajo actual no debe de estar dentro del directorio a borrar.

Sintaxis: *rmdir <dir1> <dir2> . . . <dirN>*

Donde *<dir1>* a *<dirN>* son los directorios a borrar.

Ejemplo: *rmdir /home/jcalvo/papers* borra el directorio */home/jcalvo/papers* si está vacío.

- **man.** Muestra la página de manual del comando o recurso (cualquier utilidad del sistema que no es un comando, como funciones de librería) dado:

Sintaxis: *man <commando>*

Donde *<commando>* es el nombre del comando o recurso sobre el que queremos obtener la ayuda.

Ejemplo: *man ls* muestra ayuda sobre la orden *ls*.

- **more.** Muestra el contenido de los ficheros indicados, una pantalla cada vez.

Sintaxis: *more <fichero1> <fichero2> . . . <ficheroN>*

Donde *<fichero1>* a *<ficheroN>* son los ficheros a mostrar.

Ejemplo: *more docs/notas* muestra por el terminal el contenido del fichero *docs/notas*.

- **cat.** Oficialmente usado para concatenar ficheros, *cat* también es utilizado para mostrar el contenido completo de un fichero de una vez

Sintaxis: *cat <fichero> <fichero2> ... <ficheroN>*

Donde *<fichero>* a *<ficheroN>* son los ficheros a mostrar.

Ejemplo: *cat docs/notas* muestra por el terminal el contenido del fichero *docs/notas*.

- **echo.** Envía al terminal los argumentos pasados.

Sintaxis: *echo <arg1> <arg2> ... <argN>*.

Donde *<arg1>* a *<argN>* son los argumentos a mostrar.

Ejemplo: *echo <<.Hola mundo>>* muestra la cadena *Hola mundo*.

- **grep.** Muestra todas las líneas de un fichero dado que coinciden con un cierto patrón.

Sintaxis: *grep <patrón> <fichero1> <fichero2> ... <ficheroN>*

Donde *<patrón>* es una expresión regular y *<fichero1>* a *<ficheroN>* son los ficheros donde buscar.

Ejemplo: *grep users /etc/passwd* mostrará todas las líneas en el fichero */etc/passwd* que contienen la cadena *users*.

CAPÍTULO 3. OTROS SISTEMAS OPERATIVOS

Actividad a realizar:

En este capítulo vamos a desarrollar un breve manual con la estructura básica de cada uno de los sistemas operativos relacionados. Sus ventajas, desventajas y características técnicas para la instalación. Si es posible realizamos la instalación de uno de ellos. También elaboramos el respectivo manual de instalación. Probarlo en la sesión práctica.

Aunque los sistemas operativos más comerciales y frecuentemente utilizados son Windows y Linux, es importante revisar la arquitectura de otros sistemas, que aunque menos populares son de gran utilidad en soluciones de red robustas y escalables.

Dentro de estos se revisaran los aspectos generales de sistemas operativos como Novell Netware, OS2 y VMS.

3.1 NOVELL NETWARE

3.1.1 Introducción al sistema operativo de red Novell

Novell NetWare está en el mercado desde 1983, el mismo año en que IBM introdujo la computadora personal IBM XT y el DOS 2.0 para IBM PC. Novell desarrolló originalmente NetWare para ejecutarse en un servidor basado en el microprocesador Motorola MC68000 usando configuración de red Novell S-Net. La presentación del XT de IBM y la versión 2 del DOS hizo ver a muchas empresas, entre ellas Novell, la oportunidad de desarrollo del producto. Como el código de NetWare estaba escrito en C, que es un lenguaje de los denominados "portables", Novell pudo trasladar parte del código del NetWare existente al nuevo equipo.

Como es sabido, el entorno DOS/Intel 8088 no es el mejor para ejecutar aplicaciones multiusuario, especialmente un sistema operativo multiusuario como NetWare. El BIOS (sistema básico de entradas/salidas), desarrollado para el PC original (y necesario con el DOS), está diseñado para ambientes monousuario. Novell tomó la importante decisión de dejar de lado completamente este sistema de E/S y crear un sistema operativo que funcionase de forma más efectiva en modo multiusuario. Debido a esto, NetWare se escribió específicamente para el hardware de los sistemas basados en el 8088, sin tener en cuenta el DOS y su sistema de E/S. Las dificultades de Novell estribaron en la necesidad de escribir y actualizar constantemente los controladores para ofrecer compatibilidad con el DOS a los usuarios. Estos problemas fueron solventados rápidamente usando un shell para DOS en las estaciones de trabajo. El shell es un interfaz software que permite a los usuarios de las estaciones trabajar con el DOS de forma normal, ejecutando también órdenes NetWare. El shell intercepta las órdenes de la red y

las dirige al servidor. Casi todas las aplicaciones del DOS se pueden ejecutar en el sistema operativo NetWare, gracias a su shell para DOS.

Mientras tanto, Novell siguió mejorando NetWare al ritmo de los avances tecnológicos. NetWare 286 funciona en modo protegido del procesador 80286, el más eficiente. En 1989, Novell presentó NetWare 386, el primer sistema operativo que aprovechaba al máximo las ventajas del microprocesador Intel 80386.

¿Qué es una red?

Una red es un grupo de computadoras que pueden comunicarse entre sí, compartir recursos (discos duros, impresoras, ...) y acceder a otros equipos u otras redes.

Los componentes básicos del hardware de una red son:

- Uno o más servidores
- Estaciones de trabajo
- Dispositivos periféricos
- Tarjetas de red
- Medios de transmisión

¿Qué es NetWare?

NetWare es un sistema operativo de red de computadores desarrollado por la empresa Novell. Es un conjunto de aplicaciones diseñadas para conectar, gestionar y mantener una red y sus servicios.

Una red NetWare utiliza el software NetWare para habilitar la comunicación entre dispositivos y el compartimiento de recursos.

NetWare es un conjunto de componentes software. Algunos de estos componentes sólo se pueden ejecutar desde el servidor de NetWare. Otros sólo se pueden ejecutar desde las estaciones de trabajo.

Soporte de estaciones de trabajo

A una red NetWare podemos conectar los siguientes tipos de estaciones de trabajo:

- DOS
- Windows
- OS/2
- Macintosh
- UNIX

3.1.2 Características de las distintas versiones

NetWare, Versión 2.2

La adaptabilidad de las características de NetWare 2.2 a las necesidades al mercado de hoy no son suficientes cuando se comienza a mencionar los asuntos de conectividad a que se enfrentan las compañías de hoy, administración y apoyo para múltiples protocolos, conexiones de área amplia, flexibilidad y facilidad de uso al administrador del NOS bajo escenarios de conectividad que cambian constantemente.

El NetWare 2.2 no pudo mantener el ritmo de los demás en las pruebas de ejecución que representaban tareas de redes mayores. Esto se puede comprender si se tiene en cuenta que NetWare 2.2 de 16 bits todavía se puede ejecutar en una máquina de clase AT.

NetWare 386 inicialmente sólo estaba disponible como una versión de hasta 250 usuarios. No tiene la capacidad de procesar múltiples hilos de NetWare 3.11 y 4.0, aunque puede ejecutar aplicaciones basadas en el servidor de llamadas a procesos (VAPs). Pero los VAPs se consideran como difíciles de escribir y hay pocos disponibles.

Requerimientos:

PC basada en una 286 o superior.
500K de RAM (2.5 Mb recomendados.)

NetWare, Versión 3.11

NetWare 3.11 sigue siendo utilizado bastante, es fuerte y flexible dentro de los sistemas operativos de red para las compañías pequeñas. Su única desventaja para los que necesitan una solución a nivel de empresa es que carece de un servicio global de directorios. Pero esto se puede corregir en parte con el NetWare Naming Service (NNS) que ofrece parte de los servicios distribuidos a los LANs de NetWare.

Ofrece la habilidad de compartir archivos e impresoras, velocidad, seguridad, apoyo para la mayoría de los sistemas operativos, y una gran cantidad de Hardware. Aunque tiene algunas dificultades con la administración de memoria, todavía vale la pena, pues tiene algunas otras características que lo hacen importante.

La principal atracción de un NOS de 32 bits como el que introdujo Novell, fue su diseño modular.

Los NLMs se pueden actualizar sin tener que reconstruir el NOS completo, y se pueden cargar sobre la marcha. Además, solamente los módulos necesarios se cargan en el NOS, reservando la memoria para otras funciones como el caché de discos. Una desventaja de este diseño es el uso de memoria. Los NLMs se cargan

en el hilo 0 y pueden tratar el servidor si el NLM no está escrito correctamente o si entran en conflicto con el NLM de otro fabricante. Por otra parte algunos de los módulos no desocupan la memoria cuando se descargan (Estos problemas de administración de memoria se resolvieron luego en NetWare 4.x).

NetWare 3.11 está diseñado en su mayoría para redes desde pequeñas a medianas que consisten en servidores individuales, principalmente porque sus servicios de directorios no integran a la red en su totalidad. Cada uno de los servidores mantiene una base de datos centralizada de verificación individual llamada el Bindery. El Bindery del servidor mantiene la información como los nombres de conexión, las contraseñas, los derechos de acceso y la información de impresión. Si los usuarios necesitan conectarse a más de un servidor para compartir recursos, deben hacerlo manualmente con cada servidor.

Requerimientos:

PC basada en una 386 o superior.
4Mb de RAM.
50Mb de espacio en Disco Duro.

NetWare, Versión 4.0

NetWare 4.0 ofrece la conexión simplificada de múltiples servidores, la capacidad de compartir recursos en la red y la administración centralizada en un mismo producto. La arquitectura de NetWare 4.0, es similar a la de la versión 3.11, como se mostró en la Figura 1.5, pero se han corregido y aumentado sus capacidades. NetWare 4.0 no es para todo el mundo. Determinar si en realidad se necesita un NOS tan potente depende del tamaño, la configuración y la complejidad de la LAN que se quiera formar. Algunas de las características nuevas más atractivas son el NetWare Directory Services (NDS), la compresión de archivos, la subasignación de bloques, la distribución de archivos y la administración basada en Microsoft Windows.

Netware, versiones superiores

Hoy la estrategia de computación en red de Novell es una arquitectura llamada **Sistemas abiertos netware**. Esta arquitectura tiene los siguientes objetivos:

- Permitir disponer de los servicios ofrecidos por NetWare en plataformas ampliables.
- Hacer que NetWare sea independiente del protocolo soportando los estándares importantes de la industria, como TCP/IP y los niveles de protocolo OSI.
- Ofrecer encaminamiento (routing) y redes de área amplia.

- Mantener abierta la arquitectura y ofrecer herramientas de desarrollo para crear aplicaciones que operen en un entorno distribuido de computación en red.

Novell planea implementar esta estrategia ofreciendo o soportando plataformas de servidores, arquitectura abierta, una tecnología de protocolos abierta y servicios NetWare.

3.1.3 Administración de memoria

Se agregó el apoyo de Memoria virtual (VM) al kernel NetWare 5. En las versiones anteriores de NetWare, si se tenía un servidor con 64MB de RAM, esa era toda la memoria que el servidor podía usar para cargar aplicaciones y NLMs así como la memoria de los pedidos de impresión, creación de datos de usuario, manipulación, y acceso de la aplicación. Con la Memoria Virtual, las aplicaciones del servidor y programas de NLM pueden hacer swap hacia y desde memoria y guardarse en el disco duro.

Por defecto, VM asigna un archivo de 2MB para swap que se ubica en la raíz del volumen del SYSTEM. De allí, el archivo de swap crecerá o achicará, mientras dependiendo del número de NLMs y aplicaciones del servidor que el sistema tiene en funcionamiento, y cuánta memoria principal necesita el sistema para mantener las otras aplicaciones del servidor y usuarios. Usted también puede seleccionar un volumen diferente para poner el archivo del swap. También se puede tener archivos de swap múltiples. Usa el comando de consola SWAP para cambiar el archivo del swap a un volumen diferente o agregar otro archivo de swap.

3.1.4 Administración del procesador

NetWare 6 es un sistema operativo de red de segunda generación del **Multiprocessing** de Novell. Novell introdujo esta funcionalidad con NetWare 4.x.

Esta primera tentativa fue algo limitada. Toda la funcionalidad del SO tuvo que ser concentrada en el procesador 0, el procesador por defecto en que los threads están ejecutándose cuando el proceso no es **Multiprocessing**.

Esta versión de NetWare permitió los usos que fueron escritos al estándar del **Multiprocessing** en varios procesadores con excepción del procesador 0. Pero en cualquier momento se necesitó utilizar la funcionalidad del SO (acceso al disco, transmitir en el hilo, etcétera) la petición tuvo que ser ejecutada de nuevo al procesador 0. Por lo tanto, no era una solución completa. Con el advenimiento de NetWare 5, la funcionalidad del **Multiprocessing** fue reescrita e integrada totalmente en el núcleo del OS de NetWare. Esto hizo extensa la funcionalidad del SO **Multiprocessing**. Sin embargo, todavía había algunos servicios esenciales que tuvieron que funcionar en el procesador 0. La funcionalidad tal como

conductores del LAN y conductores del disco todavía necesitó **Multiprocessing** permitido. En NetWare 6, todos los componentes son **Multiprocessing** obedientes. La cadena de acontecimientos entera, del hilo de la red a los dispositivos de almacenaje en disco duro, son **Multiprocessing**-permitidos. Así NetWare 6 provee una solución completa de servidor **Multiprocessing**.

3.1.5 Los servicios del índice de Novell (Novell Directory Services)

La visión de Novell respecto a la futura gestión de una empresa pasa por la construcción de una red global inteligente que conecte Internet, grupos de trabajo y redes corporativas en un único sistema de información orientado a las empresas, los clientes y los usuarios. Los tres elementos que hacen realidad esta red global son:

- Servicios de red que trabajan de forma inteligente para los usuarios. Estos servicios identifican a los usuarios cuando se conectan, determinan dónde están, qué necesitan y cómo trabajar de la mejor forma posible para ellos.
- Acceso universal, es decir, en cualquier momento y desde cualquier lugar se puede acceder a la red.
- Integración heterogénea que consolida los productos y dispositivos de distintos fabricantes en una única red. Asignación dinámica entre un objeto y el recurso físico al cual se refiere.

NDS está formado por una serie de objetos colocados en una estructura jerárquica con forma de árbol invertido. Una empresa puede organizar los objetos en el índice según la forma en que los usuarios acceden a los recursos y los utilizan. De esta manera acceder a un recurso es una tarea sencilla y que permite que este servicio se utilice para establecer una administración basada en reglas. La administración basada en normas permite a los administradores gestionar una rama entera del índice con una simple modificación. De esta forma se pueden conceder seguridad de acceso a toda la empresa sencilla y rápidamente, minimizando la necesidad de administrar múltiples grupos.

Lo más importante de este servicio es la transparencia en la jerarquía y la herencia a lo largo de todo el índice sin importar el número de servidores. Por ejemplo, al conceder un permiso a una rama del árbol dicho permiso lo heredan de manera automática todos los usuarios que se encuentren por debajo, ya sean diez o varios miles.

3.1.6 ADMINISTRACIÓN DE DISPOSITIVOS

Arquitectura ODI

ODI (Open Datalink Interface o Interfaz de Enlace de Datos Abierto) es una especificación definida por Novell Corporation y Apple Computer Corporation para simplificar el desarrollo de controladores de red y proporcionar soporte para

múltiples protocolos sobre un sólo adaptador de red o incluso para hacer convivir varios adaptadores de red sobre el mismo sistema operativo.

ODI proporciona a los protocolos una API (Interfaz de Programación de Aplicaciones) que permite comunicar con el adaptador de red y la convivencia de distintos protocolos simultáneamente.

La configuración de Netware con ODI está compuesta de los siguientes módulos de software entre otros:

- MLID (Multiple Link Interface Driver). Es el programa que controla al adaptador de red, especialmente preparado para la utilización de la tecnología ODI. Cada tarjeta tiene un módulo MLID distinto, que normalmente recibe el nombre del adaptador y tiene extensión COM. Así, la tarjeta NE2000 tiene un módulo MLID denominado NE2000.COM.
- LSL.COM (Link Support Layer o Legislador de Soporte de Enlace). Provee la capacidad para la convivencia múltiple de protocolos en una o más tarjetas de red. Sobre este módulo se asientan otras capas de software para habilitar la gestión de red de distintas tecnologías: IPX, TCP/IP, etc.

Por encima de estos módulos se pueden instalar otros de software, como en el caso de la configuración monolítica.

3.1.7 Servicios WEB

Con Netware 5 se incluye el servidor Web FastTrack versión 3.5 desarrollado por la empresa Novonyx, que fue fundada por Novell y Netscape con el propósito de incorporar toda la línea de productos de Netscape sobre la plataforma Netware. FastTrack es el mismo servidor Web que se puede encontrar en otros sistemas operativos.

A principios de 1998 Novell y Netscape anunciaron que su subsidiaria Novonyx se encargaría de proporcionar los servidores de Netscape SuiteSpot sobre la plataforma Netware. Todas las funciones administrativas, de ventas y soporte serán coordinadas por Novell a partir de dicha fecha.

El producto cuyo nombre oficial es "Netscape FastTrack Server for Netware 5" es un servidor WEB de alto rendimiento basado en estándares abiertos. Con él se entrega información una nuestra intranet o Internet empleando el protocolo HTTP.

UNICON. El programa con el que se realiza toda la administración de los servicios FTP, LPD, LPR y NIS se llama UNICON.NLM. Se debe ejecutar en el servidor Netware y no está integrada con el Administrador normal de Windows.

3.1.8 Recursos y servicios de red

Uno de los objetivos de una red es el de proporcionar acceso a los servicios y recursos de red. Un recurso es algo que se utiliza, como una impresora de la red o un volumen en una unidad de disco. Un servicio de red es el sistema empleado para proporcionar un recurso. La mayor parte de los recursos y servicios los proporcionan los servidores de NetWare y pueden ser compartidos simultáneamente por múltiples clientes que se encuentren en cualquier punto de la red.

Un servidor NetWare puede proporcionar una amplia gama de servicios de red, todos los cuales podrían considerarse como responsabilidad del administrador de la red:

- Servicios del Directorio Netware
- Seguridad
- Sistema de archivos
- Impresión en red
- Servicios de gestión del almacenamiento
- Servicio del sistema de gestión de mensajes

Servicios del Directorio Netware. Los Servicios del Directorio NetWare (o NDS) mantienen una base de datos con información acerca de todos los recursos de una red NetWare organizada en una estructura de árbol jerárquica llamada *árbol del directorio*. Los NDS procesan las peticiones de recursos de red que hacen los clientes: localizan el recurso en la red, verifican el cliente y lo conectan al recurso.

Seguridad. El sistema de seguridad de NetWare no es un sistema unitario, sino que está distribuido entre los muchos servicios de la red. El papel de cada sistema de seguridad es el de regular el acceso a los recursos de red. El componente fundamental de la seguridad de la red es su cuenta de usuario, que le identifica e indica qué es lo que está autorizado a hacer en la red.

Sistema de archivos. Sistema que utiliza el servidor de NetWare para organizar datos en los discos duros. Cada archivo recibe un nombre de archivo y se almacena en un punto determinado en un sistema jerárquico de archivo para que los archivos se puedan localizar rápidamente.

El sistema de archivos NetWare permite que los clientes comparten unidades de disco conectadas a los servidores de NetWare. Se puede utilizar para almacenar, compartir y utilizar aplicaciones y archivos de datos.

Impresión en red. NetWare permite a todas las estaciones de trabajo imprimir por las mismas impresoras. Se pueden conectar estas impresoras a los servidores de NetWare, a las estaciones de trabajo o directamente al cable de la red. Gracias al

servicio de impresión de NetWare podremos, desde nuestro puesto de trabajo, imprimir por cualquier impresora que esté conectada a la red.

Servicios de gestión del almacenamiento. Los servicios de gestión del almacenamiento (SMS) controlan el respaldo y la recuperación de los datos de los servidores y estaciones de trabajo. Un respaldo es una copia, un duplicado de los datos del sistema de archivos que se copia en un dispositivo de almacenamiento (generalmente, una cinta de cartucho). Se pueden recuperar y restaurar el respaldo en caso de que se degrade o destruya el original.

Servicio de sistema de gestión de mensajes (MWS). El Servicio de sistema de gestión de mensajes de NetWare (MHS) es un servicio de gestión de mensajes que permite intercambiar correo electrónico, compartir calendarios, programar los recursos y realizar otras muchas tareas.

3.2 Sistema operativo OS/2

3.2.1 Introducción al sistema operativo OS/2

Éste es un sistema operativo del que casi todo el mundo ha oído hablar, pero casi nadie ha probado.

Los objetivos principales de los diseñadores de OS/2 fueron crear un sistema operativo ideal para la automatización de oficinas, proporcionar manejadores gráficos independientes de los dispositivos, lograr que las aplicaciones tuvieran acceso directo a periféricos con gran ancho de banda, ofrecer capacidad multitarea, proporcionar un ambiente adaptado para cada programa y para sus descendientes, y ofrecer un ambiente protegido para garantizar la estabilidad del sistema.

OS/2 fue desarrollado originalmente entre IBM y Microsoft como un sucesor multiproceso del DOS para CPUs 286 y mejor, pero la versión 1.x nunca fue aceptada excepto para algunas aplicaciones específicas. Con la versión 2.0, Microsoft dejó la sociedad OS/2, e IBM promovió el OS/2 como un sistema operativo de 32-bit que requería un CPU 386 o mejor.

OS/2 fue originalmente diseñado por Microsoft con la ayuda de IBM. Desde el punto de vista de estas compañías OS/2 iba a reemplazar a MS-DOS. Esto nunca sucedió OS/2 se entregó tarde e incompleto. Aunque tenía unas ventajas obvias sobre MS-DOS, como el uso real de memoria, la ejecución en modo protegido y el soporte de multiprogramación en forma elegante, los usuarios no se interesaron en él.

3.2.2 Compendio de Versiones de OS/2

OS/2 1.0 (1987) fue originalmente diseñado por Microsoft con la ayuda de IBM. Cuando el procesador 286 era el último y mas grande chip. Desde entonces Microsoft se percató que DOS se estaba quedando atrás.

Ellos introdujeron OS/2 1.0 en 1987, el cual corre en modo texto programas que eran extremadamente poderosos.

Desde entonces sobrepasan más de los límites de DOS, usando aun comandos estilo DOS en la línea de prompt.

OS/2 1.1(1988) - 1.3 (1991) incluyó el administrador de presentación, el cual se veía como Windows 3.X.(Windows 3.X tardaría para ser mas avanzado que el Presentation Manager). El apoyo a DOS fue adicionado con algunas otras características como alta o excelente ejecución del sistema de archivos, el cual en verdad permitía nombre de archivos largos, y esto es típicamente mucho más rápido que (V) FAT y permitía volúmenes de tamaño mas largo son aquellos torpes o en cómodos programas administradores de discos.

OS/2 2.0 (1992) vio el amanecer de la siguiente generación en una interface de usuarios gráfica PCs IBM. En vez de ir a un procesador de palabras para imprimir un documento, simplemente se arrastra el icono del objeto documento al icono de impresión. OS/2 2.0 también lució uno nuevo, 32 bits, 386 basado en KERNEL alejó limitaciones como nunca antes. También OS/2, corre aplicaciones de Windows 3.0 en modo real, extendió el encanto de OS/2 como una Plataforma Integrada sobre la cual podrá correr todas sus aplicaciones.

OS/2 2.1 (1993) introdujo un sistema gráfico de 32 bits con algunas mejoras aquí y allá en velocidad, y muchos más manejadores. También introdujo el Multimedia Presentation Manager (MMPM/2) una lista standar de aplicaciones y utilitarios de Os/2. El soporte del programa Windows 3.1 (modo extendido) fue agregado a esta versión.

Fue probablemente uno de los logros mas importantes que ha tenido IBM, este usaba la copia existente de Windows en el disco duro para correr las aplicaciones de Windows, más que eso era equivalente al OS/2 2.1 de DOS.

OS/2 Warp 3.0 (1994). Marcó un hito para OS/2, después de un amplio ciclo de prueba BETA sobrevino y salió un producto altamente refinado y pulido que lució un KERNEL mucho más rápido y nuevas rutinas de intercambio las cuales incrementaron la velocidad. Se añadió el soporte para Windows de 32 bits, junto con un alto conjunto de versiones, gratis programas incluidos en el bonus pack.

OS/2 lució el frenesí de internet de 1995 con su Internet Acces Kit. La instalación se hizo mucho más sencilla y amigable, y un utilitario de desinstalación aunque rudimentario salió a escena. Una herramienta de recuperación de desastres fue incluida dentro del sistema lo que marcó el estándar el cual es

imitado por otros sistemas operativos de hoy día. Dos versiones fueron hechas OS/2 para Windows sacada en 1994 y entonces OS/2 Warp con el OS/2 versión nativa de Windows 3.1 a principios de 1995. OS/2 Warp también soporta Windows para trabajo en grupo donde OS/2 2.1 nunca lo soportó

Os/2 Warp Connect (1995), puso a OS/2 en el rango de punto a punto y estaciones de trabajo cliente OS/2 por diseño tiene incluido en el sistema operativo la gestión de redes. Sólo libera al animal enjaulado. Connect es también conocido como un sistema confiable para clientes de redes o para pequeños servidores.

OS/2 Warp Server (1997), efectivamente integró el LAN Server de IBM en OS/2 Warp. Os/2 Warp Server ha sido aclamado como el mejor sistema operativo de redes, es muy eficiente y requiere menos hardware, que sus equivalentes funcionales de NT y UNIX para dar el mismo rendimiento.

Tiene muchos usos y características de administración los cuales han hecho de él ya el favorito para muchos de los administradores de redes

OS/2 Warp 4 ó Merlin (1998), existen infinidad de rumores a cerca de Merlin. Merlin es el nombre del proyecto dado para la siguiente revisión de OS/2 Warp. Más nunca existirá una versión OS/2 sola; siempre será un servicio de cliente y redes de punto en todas las futuras versiones de OS/2. Merlin posee mejoras en (Work place Shell), incluye mejora en los multimedios, y otras mejoras principalmente apuntaron a hacer a OS/2 mucho más amigable para usar

3.2.3 Manejo de archivos en OS/2

Debido al objetivo inicial de mantener compatibilidad con DOS, las versión 1.0 de OS/2 era muy similar a la de éste sistema operativo. Posteriormente en las versiones 2.x mejoró el sistema de archivos con otras facilidades, como ofrecer dos modos de trabajo: el síncrono y el asíncrono. El modo síncrono se realiza a través del llamado a las rutinas 'DosRead' y 'DosWrite', mientras que el asíncrono se realiza por medio de 'DosReadAsync' y 'DosWriteAsync'. En el caso de que se estén ejecutando varios 'threads' de un proceso, la sincronización de las operaciones sobre archivos se puede realizar a través de semáforos con la llamada a la rutina 'DosMuxSemWait'.

Respecto a los discos duros, OS/2 permite crear varias particiones en un solo disco y mantener sistemas de archivos en cada partición con su propio "File Allocation Table" (FAT) en cada partición. A este tipo de particiones se les llama 'particiones ampliadas'. OS/2 continua usando nombres de archivos de ocho caracteres y extensiones de tres con un punto que los separa. A continuación se muestran algunas llamadas para la manipulación de archivos.

Llamada	Descripción
DosBufReset	Graba al disco los buffers del archivo
DosClose	Cierra el archivo
DosDelete	Borra el archivo
DosDevIOCtl	Establece parámetros de trabajo
DosMkDir	Crea un directorio
DosNewSize	Cambia el tamaño de archivo
DosFileInfo	Obtiene información sobre el archivo
DosSetFileInfo	Establece información del archivo
DosOpen	Abre un archivo
DosSet FileMode	Establece el modo de operación
DosRmDir	Borra un directorio vacío
DosSelectDisk	Selecciona un disco para trabajar

Como en UNIX y algunos otros sistemas operativos, OS/2 permite ser instalado en una partición de disco duro y dejar otras intactas para instalar otros sistemas operativos, dando así la facilidad de poder usar una misma computadora con diferentes sistemas operativos. OS/2 ofrece una interfaz gráfica para que el usuario trabaje, en particular ofrece un ícono para representar los archivos y una barra de menús para realizar operaciones sobre ellos como abrirlos, cerralos, copiarlos, etc. Si el usuario está acostumbrado a teclear comandos, entonces puede pedir una sesión de DOS para usar los comandos habituales de ese sistema operativo. En particular, en el ambiente de ventanas se tiene un ícono denominado "Sistema OS/2" que contiene otro ícono llamado "Unidades" y ahí existen íconos que representan el disco duro, unidades de disco flexible, etc. Para realizar copias de archivos, borrados, etc; basta con arrastrar los íconos correspondientes de/hacia el origen/destino deseado. La versión inicial de OS/2 tenía incluído el sistema Windows, pero debido a las regalías que debía pagar a Microsoft, éste fue eliminado y el usuario debe adquirirlo por separado, y configurarlo al momento de instalación.

3.2.4 Manejo de procesos en OS/2

OS/2 utiliza un esquema de planificación apropiativa, es decir, los procesos pueden ser suspendidos para darle su turno de ejecución a otro diferente. Los procesos pueden estar divididos en 'threads' que cuentan con sus propios registros, pila y contador de programa y todos los 'threads' de un mismo proceso comparten la memoria. Esto facilita la comunicación entre ellos y la sincronización. También es posible que un proceso genere un proceso hijo, en tal caso el hijo hereda todos los atributos del padre como son los descriptores de archivos abiertos, los valores en memoria, etc; prácticamente igual que el sistema operativo UNIX.

Otra facilidad de OS/2 es la facilidad de crear 'conductos' lo cual también es una función heredada de UNIX.

La calendarización de procesos o 'threads' se hace por prioridad y dándoles una intervalo de ejecución a cada proceso o 'thread'. Se manejan tres niveles de prioridades: procesos preferentes, procesos preferentes interactivos y procesos normales. OS/2 eleva a la categoría de prefentos a aquellos procesos que hacen mucha E/S.

Otra facilidad notable de OS/2 es la carga dinámica de librerías, que consiste en la generación de aplicaciones cuyas librerías no forman parte del código compilado, sino que son cargadas cuando el programa es ejecutado. Esto sirve bastante sobre todo cuando las librerías son de uso común. Como se ve, esta facilidad es parecida a las del sistema operativo UNIX SunOS.

3.2.5 Manejo de memoria en OS/2

La versión inicial de OS/2 usaba segmentación pura debido sobre todos a las restricciones de los procesadores. Pero ya que el 80386 soportaba segmentación y paginación, IBM prometió un manejo de memoria virtual más sofisticado. El algoritmo de sustitución de segmentos era el "Menos Recientemente Usado". Con el 80386 se rompió la barrera de segmentos de 64 kilobytes para ofrecer los llamados "segmentos gigantes" que podían estar formados de varios segmentos de 64k. Debido a que OS/2 debe hacer uso del modo protegido, no se permiten algunos manejadores de extensión de memoria que violan este modo de trabajo.

En particular, la versión 2.0 soporta aplicaciones que usan el modo protegido de DOS "DOS Protect-Mode Interface", el "Expanded Memory Specification" (EMS), o el "Extended Memory Specification" (XMS). Los programas que usan WINMEM32.DLL no eran soportados, ni los que acceden directamente los sectores físicos del disco duro. Las nuevas versiones manejan memoria con paginación.

3.2.6 Manejo de entrada/salida en OS/2

En OS/2 se tuvo un gran problema de diseño en este aspecto, ya que se deseaba dar compatibilidad a los programas existentes para DOS. En este aspecto, existen gran cantidad de programas de DOS que accedían directamente a algunos periféricos, incluso interceptando los vectores de interrupciones para realizar un manejo propio en la entrada/salida. Todos esos programas no son soportados en forma nativa en OS/2, sino que deben ser recreados usando una facilidad llamada "supervisor de dispositivos".

OS/2 sigue soportando la idea de "device drivers" en una forma parecida que en DOS. De hecho, algunos estudiosos de los sistemas operativos afirman que DOS se puede considerar como un sistema "microkernel" por esta característica.

Para que un proceso sea candidato a manejar un dispositivo, debe informarlo a través de una llamada a 'DosMonOpen' y 'DosMonReg'. El supervisor de dispositivos usará un modelo de productor-consumidor para enviar y recibir datos con el proceso candidato. También es factible que para un mismo dispositivo el supervisor envíe los datos a varios procesos interesados en leer de él. Los dispositivos en OS/2 se clasifican en aquellos orientados a bloques y aquellos orientados a caracteres. Los dispositivos orientados a caracteres se manejan de manera síncrona.

Los procesos también pueden indicar los permisos de los archivos y dispositivos para indicar quiénes pueden accesarlos al mismo tiempo. De este modo se consigue que los datos estén íntegros.

También existe el servicio de reloj, lo cual permite sincronizar algunos eventos, por medio del reloj del sistema que oscila 32 veces por segundo y otro que oscila millones de veces. Dependiendo de la precisión deseada se usa el reloj adecuado. Las llamadas para el reloj de mayor precisión se hacen en un área llamada "segmento de información global" por medio de la rutina "DosGetInfoSeg".

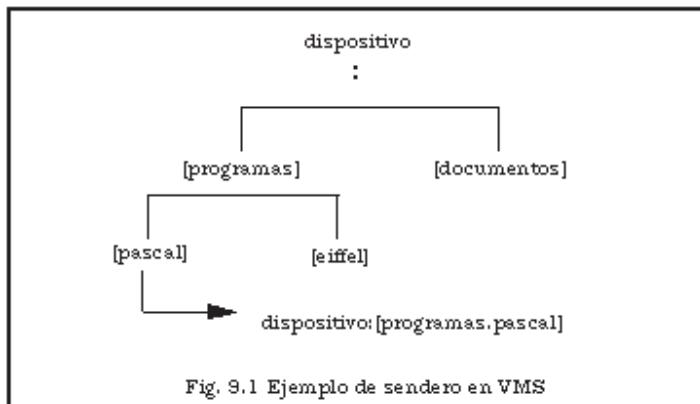
3.3 El sistema operativo VMS (Virtual Memory System)

3.3.1 Introducción al sistema operativo VMS

Es uno de los más robustos en el mercado, aunque es propietario de la compañía Digital Equipment Corporation. Actualmente con su versión OpenVMS 5.x existe para los procesadores de las máquinas VAX (CISC) y con el Alpha-chip (RISC). Ofrece un amplio conjunto de comandos a través de su intérprete Digital Command Language (DCL), utilidades de red (DECnet), formación de "clusters" de computadoras para compartir recursos, correo electrónico y otras facilidades. Es un sistema operativo multiusuario/multitarea monolítico.

3.3.2 El manejo de archivos en VMS

El sistema de archivos de VMS es jerárquico aunque la descripción de sus senderos tiene una sintaxis propia.

Gráfica 100. Sistema de archivos de VMS

Los archivos en VMS se referencian con la sintaxis “nombre.tipo;versión”, donde “nombre” es una cadena de caracteres alfanuméricos, “tipo” es la extensión del archivo y se usa generalmente para describir a qué aplicación pertenece (“pas”=pascal, “for” fortran, etc.) y “versión” es un número entero que el sistema se encarga de asignar de acuerdo al número de veces que el archivo ha sido modificado. Por ejemplo, si se ha editado tres veces el archivo “lee.pas”, seguro que existirán las versiones “lee.pas;1”, “lee.pas;2” y “lee.pas;3”. De esta forma el usuario obtiene automáticamente una “historia” de sus archivos.

La protección de los archivos se realiza mediante listas de control de acceso (Access Control Lists). Se pueden establecer protecciones hacia el dueño del archivo, hacia los usuarios privilegiados (system), hacia los usuarios que pertenecen al mismo grupo de trabajo que el dueño y hacia el resto del mundo. Para cada uno de los anteriores usuarios se manejan cuatro permisos: lectura, escritura, ejecución y borrado.

Por ejemplo, el siguiente comando:

```
$ set protection=(S:rwed,O:rwed,G:d:W:e) lee.pas
```

establece que el archivo “lee.pas” dará todos los permisos al sistema (S:rwed) y al dueño (O:rwed), mientras que a los miembros del grupo de trabajo le da permiso de borrar (G:d) y al resto del mundo permiso de ejecución (W:e).

En VMS, a través de su “Record Management System” (RMS) se obtienen las facilidades para la manipulación de archivos tanto locales como en red. En el RMS, se proveen facilidades tales como: múltiples modos de acceso a archivos para lograr accesarlos en forma concurrente y permitiendo su consistencia e integridad, establecimiento de candados automáticos al momento de apertura para evitar actualizaciones erróneas y optimización interna en las operaciones de entrada/salida al accesar los archivos. En el caso de que los archivos no son

locales, sino remotos, se utiliza internamente el protocolo llamado “Data Access Protocol” (DAP).

3.3.3 Manejo de procesos en VMS

Soporta muchos ambientes de usuario tales como: tiempo crítico, desarrollo de programas interactivos, batch, ya sea de manera concurrente, independiente o combinado.

El calendarizador VAX/VMS realiza calendarización de procesos normales y de tiempo real, basados en la prioridad de los procesos ejecutables en el Balance Set. Un proceso normal es referido a como un proceso de tiempo compartido o proceso background mientras que los procesos en tiempo real se refieren a los de tiempo crítico.

En VMS los procesos se manejan por prioridades y de manera apropiativa. Los procesos se clasifican de la prioridad 1 a la 31, siendo las primeras quince prioridades para procesos normales y trabajos en lote, y de la 16 a la 31 para procesos privilegiados y del sistema. Las prioridades no permanecen fijas todo el tiempo sino que se varían de acuerdo a algunos eventos del sistema. Las prioridades de los procesos normales pueden sufrir variaciones de hasta 6 puntos, por ejemplo, cuando un proceso está esperando un dispositivo y éste fue liberado. Un proceso no suelta la unidad central de procesamiento hasta que existe un proceso con mayor prioridad.

El proceso residente de mayor prioridad al ser ejecutado siempre se selecciona para su ejecución. Los procesos en tiempo crítico son establecidos por el usuario y no pueden ser alterados por el sistema. La prioridad de los procesos normales puede ser alterada por el sistema para optimizar overlap de computación y otras actividades I/O.

Un aspecto importante del planificador de procesos en VMS es la existencia de proceso “monitor” o “supervisor”, el cual se ejecuta periódicamente para actualizar algunas variables de desempeño y para re-calendarizar los procesos en ejecución.

Existen versiones de VMS que corren en varios procesadores, y se ofrecen librerías para crear programas con múltiples “threads”. En específico se proveen las interfaces “cma”, “pthread” y “pthread-exception-returning”. Todas estas librerías se conocen como DECthreads e incluyen librerías tales como semáforos y colas atómicas para la comunicación y sincronización entre threads. El uso de threads sirve para enviar porciones de un programa a ejecutar en diferentes procesadores aprovechando así el multiproceso.

Servicios del Sistema para el Control de Procesos

- **Crear un proceso.** El servicio de creado de sistema permite a un proceso crear otro. El proceso creado puede ser un subproceso o un proceso completamente independiente (Se necesitan privilegios para hacer esto).
- **Suspender un proceso.** Esto es que le permite a un proceso suspenderse a sí mismo o a otro (también necesita tener privilegios).
- **Reanudar un proceso.** Permite a un proceso reanudar a otro si es que este tiene privilegios para hacerlo.
- **Borrar un proceso.** Permite que se borre el proceso mismo o a otro si es que es un subproceso, o si no tiene que tener privilegios de borrado.
- **Dar prioridad.** Permite que el proceso mismo se ponga prioridad o a otros, para el calendarizador.
- **Dar el modo de espera.** Permite que el proceso escoja de dos modos: el modo por default es cuando un proceso requiere un recurso y está ocupado y espera a que esté desocupado, y el otro modo es cuando está ocupado el recurso, el proceso no espera y notifica al usuario que el recurso no se encuentra disponible en ese momento en lugar de esperar.
- **Hibernar.** Es cuando un proceso se hace inactivo pero está presente en el sistema. Para que el proceso continúe necesita de un evento para despertar.
- **Wake.** Esto activa a los procesos que están hibernando.
- **Exit.** Es cuando se aborta un proceso.
- **Dar nombre al proceso.** Este puede dar un nombre al proceso mismo o cambiarlo (el PCB contiene el nombre).

3.3.4 Manejo de memoria en VMS

El sistema operativo VMS utiliza un esquema de manejo de memoria virtual combinado de segmentación paginada. Lo novedoso en VMS es que usa un doble esquema de paginación cuando las páginas se van a intercambiar de memoria RAM hacia disco duro. En primer lugar, cuando una página necesita cargarse a RAM ésta se carga junto con varias páginas que están adyacentes, justificando esto por medio de la teoría del conjunto de trabajo que especifica que es muy probable que las referencias a memoria en el futuro inmediato caerán precisamente en esas páginas. De este modo, se tiene un doble algoritmo: al hecho de cargarse las páginas cuando se necesitan se le llama “paginación por demanda” y al hecho de traerse las otras páginas del conjunto de trabajo por anticipado se le llama “paginación anticipada”.

3.3.5 El manejo de entrada/salida en VMS

En VMS, se usan nombres “lógicos” para describir a los dispositivos existentes en el sistema. Un concepto importante tanto en archivos como en dispositivos es el “User Identification Code” (UIC) que permite establecer protecciones adicionales a

los ACL. En los dispositivos se manejan cinco tipos de permisos: leer, escribir, ejecutar, borrar y controlar. No todos los permisos se aplican a todos los dispositivos. El permiso de “control” no se maneja explícitamente sino que se otorga por omisión al dueño y al sistema. Los permisos de los discos, unidades de cinta y otros dispositivos son establecidos por el administrador del sistema.

Los dispositivos reciben nombres “lógicos”, por ejemplo, para una unidad de cinta el nombre puede ser “MTA0”.

Maneja System Interface (SCSI) que son ampliamente usados en diversas plataformas. El intercambio de datos entre la unidad central de proceso y los periféricos se lleva a cabo a través de los “buses” normalizados UNIBUS y MASSBUS.

CAPÍTULO 4. ARQUITECTURA DE LAS COMUNICACIONES

Actividad inicial:

En este punto, vamos a establecer un comparativo entre las arquitecturas aquí mencionadas. A nivel de capas, costos, servicios, equipos, sistemas operativos que pueden manejar, etc.

Es importante resaltar ventajas y desventajas de cada una, y al final proponer cuál sería la arquitectura ideal para un sistema de red estándar.

La arquitectura de las comunicaciones es una estructura organizada jerárquicamente con el fin de permitir el intercambio de datos entre niveles lógicos semejantes en distintas máquinas o terminales de la misma o distinta red.

Es hora de conocer y/o recordar los diferentes modelos arquitectónicos propuestos para la implementación de una red. Recordemos que esto es importante a la hora de seleccionar y configurar las características de un sistema operativo cualquiera para una red de computadores.

4.1 El modelo arquitectónico de capas de red

En una arquitectura de red basada en capas, existen varios niveles de capas con interfaces entre ellas. Las interfaces proporcionan los puntos de acceso a los diferentes servicios que cada capa provee; la primera capa no tiene otra por debajo a quien solicitar servicios, ésta se encarga de operar con los medios de transmisión.

Una ventaja de esta arquitectura es que es poco sensible a los cambios tecnológicos que se producen por evolución en las funciones y en los servicios de las redes lo que las hace enormemente flexibles.

El proceso de comunicación se produce entre capas equivalentes de dos host distintos. La información va descendiendo por la estructura de capas del host emisor hasta llegar al nivel más bajo, de donde pasa al host receptor y aquí se inicia el viaje ascendente hasta llegar a la capa equivalente en el host de destino.

La capa N de un host emisor se comunica con la capa N de un receptor a través de un protocolo que enmascara el proceso desencadenado en las capas de nivel inferior haciéndolo transparente. La capa 1 opera con transmisiones en el nivel físico, es decir con señales; el resto de las capas opera con comunicaciones.

4.2 El modelo de referencia OSI

OSI es el nombre de una arquitectura de capas para redes de ordenadores y sistemas distribuidos propuesto por la ISO como estándar de interconexión de sistemas abiertos.

4.2.1 La estructura de capas en OSI

El modelo OSI propone siete capas o niveles diseñadas teniendo en cuenta los siguientes factores:

- Una capa se identifica con un nivel de abstracción.
- Cada capa debe tener una función perfectamente definida.
- La función de cada capa debe elegirse de modo que sea posible la definición posterior de protocolos.
- Se disminuirá al máximo el flujo de información entre las capas a través de los interfaces.
- Las capas serán tan numerosas como sea necesario para que dos funciones muy distintas no tengan que convivir en la misma capa.

4.2.2 La comunicación entre las capas

La comunicación se realiza a través del interface, y se da por un sistema de llamadas y respuestas denominado primitivas. Cada servicio es el conjunto de primitivas que cada capa ofrece a través de su interface a la capa superior y está nominado por un SAP que lo identifica únicamente dentro de cada interface.

OSI define cuatro primitivas:

Primitiva	Nombre OSI	Significado
Solicitud	.request	Una entidad solicita que un servicio realice un trabajo para ella.
Indicación	.indication	Una entidad es informada de que ha ocurrido un evento.
Respuesta	.response	Una entidad responde con una primitiva a un evento producido anteriormente
Confirmación	.confirm	Una entidad es informada acerca de una solicitud efectuada anteriormente

La primitiva de un servicio se construye escribiendo el nombre del servicio(en mayúsculas) seguido por un punto y la primitiva fundamental. Ejemplo CONNECT.request para hacer una petición.

4.2.3 Tipos de servicios definidos en OSI:

Existen dos tipos de servicios:

Servicios orientados a la conexión. Requieren el establecimiento inicial de una conexión y la ruptura o liberación final de la misma, en donde se produce el intercambio de datos del usuario. Los bloques de datos se reciben en el mismo orden en que fueron emitidos y todos los paquetes siguen la ruta conseguida en la conexión, por ejemplo el servicio telefónico.

Estos tienen dos variantes:

- **Secuencia de mensajes:** Se establecen fronteras que definen y determinan cada mensaje.
- **Secuencia de bytes:** no hay contorno entre los mensajes. Cada mensaje es una secuencia de caracteres dejando al receptor la responsabilidad de su interpretación.

Servicios sin conexión. Ofrecen la capacidad de comunicación sin realizar una conexión con el destinatario. Se envían paquete de datos con la dirección de destino, confiando en que la red conduzca los datos por la ruta adecuada. En algunos casos, el receptor debe enviar acuse de recibo al emisor, por ejemplo el sistema postal.

Existen varios tipos de servicio sin conexión:

- **Servicio de datagrama sin confirmación:** no necesita confirmación por parte del receptor, ejemplo protocolo IP
- **Servicio de datagrama con confirmación:** se envía confirmación al emisor, ejemplo correo electrónico con acuse de recibo.
- **Servicio de petición y respuesta:** se basa en que a cada petición le sigue una respuesta. Ejemplo, en una base de datos.

4.2.4 Los niveles OSI orientados a la red

	NIVEL	FUNCTION	Modo de Transporte
Aplicación o usuario	Aplicación	Se definen los protocolos que utilizarán las aplicaciones y procesos de los usuarios. La ISO referencia 5 grupos de protocolos: Grupo 1. Protocolos de gestión del sistema. Grupo 2. Protocolos de gestión de la aplicación. Grupo 3. Protocolos de sistema Grupo 4 y 5. Protocolos específicos para aplicaciones	APDU
	Presentación	Se ocupa de la sintaxis y de la semántica de la información, es decir, investiga en el contenido informativo de los datos; comprime los datos para que las comunicaciones sean menos costosas y encriptación de la información	PPDU

	NIVEL	FUNCIÓN	Modo de Transporte
	Sesión	Permite el dialogo entre el emisor y el receptor, estableciendo una sesión en la cual se puede dar un transporte de datos ordinario. Se realiza en dos etapas: ➤ Establecimiento de sesión y creación de un buzón. ➤ Intercambio de datos entre los buzones Determina si la comunicación será bidireccional o simultanea	SPDU
	Transporte	Acepta los datos de la capa de sesión, los fracciones para que sean aceptados por la subred y se asegura que de que llegará al nivel de trasporte del destinatario	TPDU
Red (Subred)	Red	Se ocupa de la subred, su función es la del encadenamiento, o sea, elegir la ruta mas adecuada para que el paquete llegue a su destino el cual esta identificado por una dirección, además trata la congestión y la resolución de problemas generados por redes heterogéneas.	PAQUETE
	Enlace	Establece una línea de comunicación libre de errores que se produzcan en la recepción de tramas, eliminar tramas erróneas, solicitar retransmisiones, adecuar el flujo de datos, etc.	TRAMA
	Físico	Define las características mecánicas, eléctricas, funcionales y de procedimiento para establecer y destruir conexiones entre dos equipos. Garantiza la compatibilidad de los conectores	BIT

Un ordenador puede soportar múltiples aplicaciones simultáneas que solicitan servicios de comunicación a la capa de transporte. A su vez, la capa de transporte debe solicitar servicios a la subred con el fin de elegir la que sea más necesaria, la ruta más conveniente y el fraccionamiento de datos más adecuado. Muchas comunicaciones de alto nivel pueden ser ejecutadas por múltiples transmisiones de bajo nivel. Sin embargo el nivel de la capa de transporte tiene que ser común.

4.3 Otras arquitecturas y redes

Siendo el modelo de referencia OSI teórico, no hay ninguna red 100% OSI, y existen otras arquitecturas que han evolucionado, siendo compatibles con OSI. Dentro de ellas tenemos: SNA de IBM, DNA de DEC y ARPANET

4.3.1 La arquitectura SNA de IBM

SNA (system network architecture). Red propia de IBM. El modelo OSI se configuró a partir de SNA, de donde toma el número de funciones aproximadas de sus capas.

La primera versión comenzó en 1974 para gestionar redes en forma de árbol con un solo host al que se conectaban sus terminales. La segunda versión en 1976, en ella se permitían varios host con sus respectivos árboles pudiendo establecer comunicación entre ellos. En 1985 se incluyeron el resto de las topologías y relaciones de área local.

SNA está constituido por un conjunto de máquinas conectadas a la red y llamadas nodos, que se pueden denominar como terminales, controladores, procesadores frontales y los hosts.

Cada uno de estos nodos tienen una NAU(network address unit) unidad de direccionamiento de red, que es el software por el que un proceso puede llegar a utilizar la red. Hay varios tipos de NAU. EL conjunto de hardware y software controlado por una NAU de tipo SSCP es lo que se llama dominio en SNA.

El número de capas es igual al de OSI, pero no hay correspondencia exacta entre ellas.

Arquitectura SNA

La idea que se perseguía al diseñar esta arquitectura de red era lograr unificar los muy diversos protocolos y paquetes de comunicaciones que poseía la empresa, con intención de ofrecer una imagen de marca más coherente a sus clientes, aunque manteniendo compatibilidad con los productos instalados.

En una red SNA una UCP (Unidad Central de Proceso) se conecta a la red mediante una UCT (Unidad Central de Transmisión), que puede estar integrada en la UCP. Una UCT puede soportar desde unidades a millares de circuitos, por su estructura modular.

Los usuarios de un Sistema de Comunicación SNA se denominan Usuarios Finales (UF). Un usuario final puede ser tanto una persona que opera en una estación de trabajo como el programa de aplicación que esta utilizando. Estos usuarios finales se encontrarán situados en los puntos de Entrada y Salida de la red. Las únicas entidades que utilizan la red son los usuarios finales.

Además se pueden discernir a los siguientes componentes:

- **Nodos.** Conjunto de componentes hardware y software que ejecutan las funciones de los niveles de la arquitectura SNA.
- **Enlaces.** Que conectan los nodos adyacentes y que pueden, a título de ejemplo, ser canales, líneas o enlaces microondas.
- **Unidades Direccionales de la Red.** Conocidos por las siglas NAU.

NAU's

Los usuarios finales (UF) ven la red a través de unas entidades llamadas NAU (Network Adressable Unit), que permite que los usuarios finales enviar datos a través de la red e interaccionar con los operadores de la red para desarrollar las funciones de control y gestión de la misma. El primer nivel de abstracción proporciona un puerto de entrada a la red.

Las NAU's pueden comunicarse a través de la red y se identifican por su:

- Dirección exclusiva en la red.
- Nombre simbólico exclusivo.

Las NAU's son recursos controlados por el sistema de comunicaciones y proporcionan funciones para:

- Sincronizar las comunicaciones entre usuarios finales.
- Gestionar los recursos de cada nodo.
- Controlar y gestionar la red.

Una conexión en SNA recibe el nombre de una sesión. Una sesión es una asociación lógica entre los usuarios finales a través de las NAU's. SNA define los siguientes tipos de NAU:

- Unidades Lógicas (LU)
- Unidades Físicas(PU)
- Puntos de Control de los servicios del Sistema (SSCP)

1. LU

Es la NAU la que proporciona las puertas (y direcciones) a los usuarios finales. Un usuario final puede estar representado por una o más LU, y también una LU puede estar representada a uno a más usuarios finales. Una LU se reconoce por su nombre simbólico en la red y su dirección. Las LU gestionan el intercambio de datos entre usuarios finales, actuando de intermediarios entre el usuario final y la red.

Dentro de una sesión LU-LU, las capacidades de ambas no son las mismas, pudiéndose distinguir entre LU primaria y LU secundaria:

LU Primaria (PLU)

- Establece la Sesión
- Gestiona la Sesión
- Finaliza la Sesión

LU Secundaria (SLU)

- Aceptar o rechazar la petición de establecimiento de sesión
- En ocasiones puede negociar los protocolos de la sesión.

Dichos roles se establecen en el momento de la comunicación. Algunas LU como las de los terminales únicamente pueden ser secundarias, mientras que otras LU como las de los programas de aplicación pueden ser tanto primarias como secundarias.

Se han definido varios tipos de LU como son LU0, que es la básica, LU" para terminales remotas tipo 3270 y LU 6.2 para comunicaciones entre programas transaccionales distribuidos.

2. PU

Cada nodo contiene una PU para gestionar los enlaces que conectan el nodo a los adyacentes y sus propios recursos. La implantación se suele realizar tanto en hardware como en software, dependiendo de las funciones.

Existen los siguientes tipos de PU dependiendo de las funciones que desempeñan en la arquitectura SNA:

PU tipo 5. Nodo de UCP. Contiene servicios de unidad física del nodo host y el SSCP. Soporta direcciones SNA completa, grupos de transmisiones, rutas explícitas y rutas virtuales. Contiene un SSCP, una PU y un numero variable de LU, generalmente asociados a programas de aplicación. Proporciona servicios de sesión y ruta.

PU tipo 4. Nodo controlador de Comunicaciones (UTC). Soporta las mismas funciones que las PU T5 con la excepción de que no contienen SSCP.

PU tipo 2. Nodo periférico. Soporta una dirección SNA limitada y ninguna de las funciones de la PU y T4. Nodos de este tipo son los controladores de terminales.

PU tipo 2.1. Nodo periférico. Es un nodo que reside en los nodos de los sistemas distribuidos, estos no necesitan ser controlados por un SSCP.

PU tipo 1. Nodo Periférico. Soporta un controlador no inteligente. Los servicios de esta unidad se implantan en el nodo controlador de comunicaciones.

Cada nodo de la red posee al menos una unidad física. Reside la PU en dicho nodo y le representan en el entorno de la red. Los servicios de la unidad física del nodo gestionan los aspectos físicos del mismo. Se comunican con el SSCP para enviar información de aspectos como Error/Trafico.

3. SSCP(System Services Control Point)

Las funciones del SSCP son las siguientes:

- Controlar los recursos del sistema de comunicaciones. El SSCP es el encargado de supervisar el dominio del sistema de comunicaciones al que pertenece.
- Gestionar es establecimiento de las sesiones. Controla el acceso de los usuarios finales a la red permitiendo un control de la gestión de la red. Interrelaciona la LU para ejecutar funciones como conexión y desconexión de aplicaciones.
- Recibe mandatos del operador
- Envía mensajes al operador

El SSCP reside en una UCP. Un SSCP gestiona un dominio de la red. Si la red cuenta más de un SSCP se tiene una red de múltiples dominios. Los dominios deben ser disjuntos. Todos los nodos y enlaces han formado parte de un único dominio. Un dominio es el conjunto de nodos controlados por un SSCP (nodo de tipo 5).

Nodo SNA

Es un elemento de la red en el que se implantan la función PU. Todo nodo tiene su PU y solo una. En un nodo, sin embargo, pueden conectar más de una LU.

Existen dos principales tipos de nodos en SNA:

1. Nodo subarea. Constituye los nodos tipo 4 o 5. el nombre lo reciben debido a que las direcciones de la red que lo identifican utilizando una parte de la dirección que es la dirección de la subarea. Un nodo controlador de comunicaciones controla los enlaces y estaciones de trabajo que están conectadas a él. Existen dos clases de nodos subarea:

- **De clase procesador.** Contiene un método de acceso de telecomunicaciones y proporciona las funciones de control y gestión de red.

- **De clase de controlador:** que contiene los programas de control de la red. Proporcionan los servicios de: encaminamiento, control de flujo y control de enlace de datos en la red.

2. Nodo Periférico. Todos los demás tipos de nodo. Un nodo periférico es la fuente o destino de datos.

Subarea

Una subarea es un concepto topológico. Todos los elementos de la red asociados con una UCP con una UCT constituyen una subarea . el conjunto de Pus y LU que están conectados físicamente a una PU tipo 4 o 5. Cada subarea tiene un numero único en la red SNA, el cual junto con la dirección única del elemento dentro de dicha subarea formaran la dirección completa en la red SNA.

Los nodos periféricos no necesitan la dirección completa SNA debido a que únicamente pueden formar parte de la subarea. Los servicios de la PU en el nodo subarea realizan la conversión de direcciones, de manera que solo envía una dirección de elemento al nodo periférico. Esta función recibe el nombre de conversión de función frontera.

Niveles SNA

No existe una correspondencia precisa entre los niveles SNA y los niveles OSI. Además del usuario final (que corresponde al nivel de aplicación) se define otros seis niveles que pueden agruparse en niveles de NAU y niveles de Red.

Niveles de NAU

- Servicios de NAU, que han experimentado notables transformaciones desde la definición de SNA, para adaptarse a las nuevas demandas de servicios. Comprenden los servicios de presentación, así como de servicios específicos como los servicios de transacción que define un marco para transacciones distribuidas o distribución de documentos.
- Nivel de control de flujo de datos para regular el flujo de Transmisiones/Recepción de usuario y el flujo de Peticiones/Respuestas.
- Nivel de control de transmisión para coordinar la transmisión de la sesión incluyendo la secuencia de numeración.

Niveles de Red

- Nivel de control de caminos (o de trayectos) que encaminan los mensajes a sus destinos.

- Nivel de enlace de datos que controlan el flujo de Datos en el Enlace.
- Nivel físico traduce los datos para adecuar al medio electrónico de transmisión.

Procesos distribuidos en SNA

La descripción realizada de los conceptos fundamentales de SNA corresponde a una concepción jerárquica de las redes de ordenadores, tal como era normalmente a finales de los años setenta, cuando se definieron los elementos arquitectónicos de SNA. La aparición de los ordenadores personales aceleró el despliegue del proceso distribuido a partir de mediados de los años ochenta. Con la distribución de capacidad de proceso en diversos nodos de la red, no es eficaz.

4.3.2 La arquitectura DNA de DEC

DNA (digital network architecture) es la arquitectura de red compuesta por DEC (digital equipment corporation). Consta de siete capas semejantes a las de OSI.

Relación de las capas entre las arquitecturas OSI, SNA y DNA

SNA DE IBM	OSI	DNA DE DEC
Servicio de transacción	Aplicación	Usuario
Administración de funciones	Presentación	Gestión de red
Control de flujo	Sesión	Sesión y control de red
Control de transmisión	Transporte	Extremos de comunicaciones
Control de rutas	Red	Encaminamiento
Enlace	Enlace	Enlace
Físico	Físico	Físico

4.3.3 La arquitectura de ARPANET

Arpanet no sigue el modelo OSI. Tiene protocolos equivalentes a lo que en OSI serían la capa de red y de transporte. Los más conocidos son:

IP (Internet protocol). Protocolo entre redes. Protocolo sin conexión diseñado para la interconexión de redes WAN y LAN.

TCP (Transmission control protocol) protocolo de control de transmisión. Protocolo orientado a la conexión equivalente en OSI a la capa de transporte en cuenta a su función, aunque difiere de su formato.

Entre las capas de presentación y sesión, ARPANET no tiene protocolos, pero en la de aplicación si hay varios. Los más conocidos son:

FTP (files transfer protocol). Protocolo de transferencia de ficheros de un ordenador a otro.

SMPT (*simple Mail Protocol Transfer*). Protocolo de simple de transferencia de correo electrónico a través se la red.

TELNET. Protocolo de conexión remota utilizado para conexiones remotas gestionadas como terminales virtuales.

ARPANET se ha convertido un estándar de hecho, multiplicando su utilización debido al Internet.

BIBLIOGRAFÍA

- ALCALDE, Eduardo. MORERA, Juan. PEREZ-CAMPANERO, Juan A. (1994). *Introducción a los Sistemas Operativos. Serie Informática de Gestión.* México: Editorial Mc Graw Hill.
- BARRETO ROA, Julio Humberto. (2001). *Sistemas Operativos. Guía de estudio.* Bogotá: Editorial UNAD.
- CALDERA (2003). *Kit de recursos. Unifying Unix Whit Linux For Business.*
- CAÑAS, Javier. Documento pdf: Sistemas Operativos. Catorce capítulos (1999).
- CARRETERO PEREZ, Jesús, GARCIA CABALLEIRA, Félix, ANASAGASTI, Pedro de Miguel, PEREZ COSTOYA, Fernando (2001). *Sistemas Operativos. Una visión aplicada.* Madrid: Mc Graw Hill.
- FLYNN, Ida M, MCCHOES, Ann McIver.(2001) *Sistemas operativos. Tercera Edición.* México: Editorial Thomson Learning.
- GUARQUIN, Margarita y QUIROGA, Edgar (2005). Módulo ensamble y mantenimiento de computadores. Bogotá: UNAD.
- RAYA, Laura, ALVAREZ, Raquel, RODRIGO, Víctor. (2005). *Sistema Operativos en entornos Monousuario y Multiusuario.* México: Alfaomega, Ra-Ma.
- RUEDA, Francisco. (1989). *Sistemas Operativos.* Santafé de Bogotá: Mc Graw Hill.
- SILBERSCHATZ, Avi, GALVIN, Peter, GAGNE, Greg. (2002). *Sistemas Operativos.* México: Editorial Limusa Wiley.
- STALLING, William. (2001). *Sistemas operativos. Cuarta edición,* México: Prentice Hall.
- TACKETT, J. (2003). *Edición especial Linux.* México: Prentice Hall
- TANENBAUM, S. Andrew, WOODHULL, Albert S. (1997). *Sistemas Operativos. Diseño e implementación.* México: Prentice Hall.

DIRECCIONES WEB

<http://www.tau.org.ar/base/lara.pue.udlap.mx/sistoper/>

<http://www.itver.edu.mx/so1/>

<http://www.itver.edu.mx/so2/>

<http://os.matiu.com.ar/>

<http://os-matiu.dreamhost.com/classes/clase1.html>

<http://www.iespana.es/canalhanoi/so/>

http://server2.southlink.com.ar/vap/sistemas_operativos.htm

<http://www.inei.gob.pe/web/metodologias/attach/lib616/INDEX.HTM>

<http://www.itq.edu.mx/vidatec/maestros/sis/mnogues/Unidad1.htm>

<http://www.cs.virginia.edu/~knabe/iic2332/notes01.html>

http://www.mundotutoriales.com/tutoriales_sistemas_operativos-mdtema56.htm

<http://www.microsoft.com/spain/technet/recursos/articulos/domcntrl.mspx>

<http://www.microsoft.com/spain/technet/implantacion/default.mspx>

<http://www.microsoft.com/latam/technet/productos/windows/windowsserver2003/>

<http://comala.escom.ipn.mx/proyecto/temarios/tercero/sistope.html>

http://lara.pue.udlap.mx/sist_oper/index.html

Linux:

www.caldera.com

www.conectiva.com.co

www.debian.com

www.gnu.org

www.linux.org

www.lucars.org

www.mandrake.com

www.redhat.com

www.suse.com

http://linux.ciberaula.com/articulo/introduccion_practica_linux/#paquetes_linux

Descargas Linux en español:

<http://www.ibiblio.org/pub/Linux/>

<http://www.ibiblio.org/pub/Linux/docs/LuCaS/>

<http://www.ibiblio.org/pub/Linux/docs/LuCaS/htmls/manuales.html>

<http://www.ibiblio.org/pub/Linux/docs/LDP/install-guide/translations/es/>

<ftp.cdrom.com>