UNIVERSIDAD AMERICANA DE ACAPULCO

EXCELENCIA PARA EL DESARROLLO

FACULTAD DE INGENIERÍA EN COMPUTACIÓN
INCORPORADA A LA UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO.

UN MODELO DE ENSEÑANZA DE SISTEMAS OPERATIVOS

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN

P R E S E N T A:

MIGUEL ANGEL FUENTES CASTAÑÓN

DIRECTOR DE TESIS

ING. GONZALO TRINIDAD GARRIDO

ACAPULCO,GRO

AGOSTO DE 2004

INTRODUCCIÓN

Los sistemas operativos son la parte más importante en cualquier sistema de cómputo. Actualmente no existe, en la Universidad Americana de Acapulco, una herramienta didáctica que apoye a los docentes en la enseñanza de sistemas operativos.

Es por ello que surge este trabajo, ya que es necesario enseñar a los estudiantes de licenciatura la forma en que trabajan los sistemas operativos con una profundidad mayor a la que estamos acostumbrados hasta ahora.

En el capítulo 1, se dará un panorama general de los sistemas operativos, su concepto, un poco de su historia, funciones, objetivos y tipos. Además, se hace un comparativo de los que actualmente existen, tomando en cuenta el punto de vista de un administrador de sistemas y el de la ingeniería.

En el capítulo 2, se trata la parte más importante de un sistema operativo: los procesos. Se mencionan algunos conceptos básicos, se toma como referencia el modelo de cinco estados con el cual se hacen todas las posibles combinaciones y se toma como referencia para realizar el prototipo que en el último capítulo se expone. También se hace una descripción de los procesos mencionando los atributos necesarios para el manejo de los mismos.

En el capitulo 3, se ha planteado la manera como está relacionado el hardware con el sistema operativo. Se toma como ejemplo el planteamiento de niveles del profesor Tannenbaum para explicar cómo se da la relación entre el hardware y el software. Además, utilizamos la arquitectura Intel para describir algunos de los registros que son más importantes para el sistema operativo. Por último, desarrollamos paso a paso el proceso de encendido de una computadora hasta que el sistema operativo es montado.

El prototipo está descrito en el capitulo 4, utilizando una herramienta multimedia, como lo es Flash; se muestra de manera clara y sencilla la forma en que interactúan los procesos; esto se realiza basándonos en el modelo de cinco estados descrito en el capítulo tres y utilizando algunos conceptos descritos en el resto de esta investigación. Además, hacemos mención de algunas herramientas que existen actualmente y que sirven de apoyo a la docencia de sistemas operativos.

PLANTEAMIENTO DEL PROBLEMA

La Facultad de Ingeniería en Computación, de la Universidad Americana de Acapulco, no cuenta con las herramientas didácticas necesarias que apoyen al docente en la enseñanza de los sistemas operativos. Esto se ha venido observando a través de las generaciones, ya que en el campo laboral muchos de los estudiantes ven obstruida su labor cuando se encuentran un problema con una computadora y lo irónico es que no saben la manera de atacarlo, siendo la causa principal

la falta de práctica; la mayoría de los casos están relacionados con el sistema operativo o el hardware de la máquina.

En la actualidad existen herramientas que permiten al alumno, al maestro y a cualquier persona con conocimientos básicos de computación, aprender cómo esta formado un sistema operativo. Estas herramientas nos permiten realizar prácticas que apoyen a nuestra formación en el trato con los sistemas operativos y a la solución general de problemas que pudieran presentar las computadoras.

Al final de esta investigación se pretende, además de generar un prototipo que muestre la manera en que interactúan los procesos, crear en el alumno una idea más amplia y transparente de la estructura de un sistema operativo además de darle una visión, de carácter práctico, acerca del funcionamiento de una computadora.

JUSTIFICACIÓN

El avance acelerado de la tecnología exige que quienes tengan la responsabilidad de manejarla, estén pensando permanentemente cómo aprovechar los beneficios que brinda el desarrollo de la computación.

Si bien es cierto, Guerrero es un estado con escasos recursos, se deben buscar soluciones qué permitan acercar la tecnología a la mayor cantidad de población, y que mejor manera que el ámbito universitario donde se forman las futuras generaciones.

El progreso de la región no solo lo podemos sustentar en el turismo, sino que también debemos de aprovechar su recurso humano y por ello, se debe proporcionar a la comunidad académica y científica, los medios que le permitan adquirir la más variada gama de conocimientos, aprovechando los recursos y las fuentes disponibles a nivel nacional e internacional.

Es por esto que, se vuelve necesario contar con jóvenes profesionales de la carrera de Ingeniero en Computación con los conocimientos sólidos, los cuales no sean simplemente teóricos, sino prácticos, con los cuales, sean capaces de identificar fácilmente soluciones para los múltiples problemas que pueda presentar la variada gama de sistemas operativos y plataformas de hardware que existen en la actualidad.

En este contexto el papel de la Universidad Americana de Acapulco se vuelve fundamental ya que es la encargada de formar a estos profesionales capaces de iniciar, mantener y proyectar el conocimiento de esta área para responder a nuevos y cambiantes retos.

OBJETIVOS

- Desarrollar un prototipo que sirva de apoyo a la enseñanza de sistemas operativos.

- Hacer una reseña de la evolución de los sistemas operativos.

- Presentar una comparativa de sistemas operativos.

- Mostrar cómo maneja el sistema operativo los procesos.

- Enseñar la manera cómo se relaciona el sistema operativo con el hardware.

- Hacer una descripción del proceso que se lleva a cabo para montar un sistema operativo.

- Describir el proceso de encendido de una computadora.

HIPÓTESIS

Debido a que el sistema operativo es la parte más importante de cualquier sistema de cómputo, y tomando como base que la Universidad Americana de Acapulco no cuenta con las herramientas necesarias que apoyen al personal docente en la enseñanza de los sistemas operativos, se vuelve necesario enseñar a los estudiantes de licenciatura, con la mayor profundidad posible, la manera en que un sistema operativo está construido y cuál es su relación directa con el hardware. Esto les proporcionará más y mejores herramientas para poder resolver problemas de la vida cotidiana.

Entonces, es posible mostrar el funcionamiento de un sistema operativo para la formación de estudiantes de licenciatura a través de un prototipo interactivo que muestre paso a paso, y de la manera más simple, la forma en que interactúan los procesos, sirviendo como un apoyo extra a las herramientas que ya existen, y que a través de la

práctica generarán un concepto más claro de los sistemas operativos y la computadora.

DEDICATORIA

- A mi madre Cristina Castañón Nava.

  Por todo el apoyo moral y sentimental.

- A mi padre Miguel Angel Fuentes Memije.

  Por inculcarme todo el tiempo el hábito de saber y conocer más.

- A mi hermana Alma.

  Por ser mi compañera todos estos años.

- A mi hermanita Yuri.

  Por reirte siempre de mis tonterías e interesarte siempre por mis cosas.

- A la memoria de mi amigo Alfonso Toledo Figueroa.

  Estés donde estés.

Por que no encuentro ninguna otra manera de agradecerles todo el apoyo que me han brindado, y por enseñarme que esta vida es de lucha y superación constante. Quiero expresarles que todo lo que soy es por ustedes. Gracias por estar conmigo en las buenas y en las malas.

Los llevo en el corazón.

Miguel Angel Fuentes Castañón

AGRADECIMIENTOS

- A la Universidad Americana de Acapulco.

  Por permitir que realizará mis estudios.

- Al Rector C. P. Israel Soberanis Nogueda.

- A la Facultad de Ingeniería en Computación.

  Por brindarme el apoyo necesario en el transcurso de la carrera.

- Al Ing. Gonzalo Trinidad Garrido director de la facultad.

  Gracias por su experiencia, su apoyo, y sobre todo por ser mi guía para la culminación de ésta tesis.

- Al Ing. Javier Saavedra Lluck.

  Por ser mí maestro, mi amigo, por su conocimiento, experiencia, y por todo el apoyo brindado.

- A la Lic. Julieta Álvarez.

  Por su apoyo y motivación.

- A mis maestros.

  Por permitir que aprendiera de ustedes.

- A mis amigos.

  Nelson, Pablo, Tomas, Orlando, José Luis, Gustavo, Zeus, Arnulfo, Luis y Enrique quienes aportaron un granito de arena en la elaboración de esta tesis y sobre todo gracias por acompañarme durante esta carrera.

- A ti Adriana.

  Por tu apoyo, paciencia, comprensión y cariño que me has dado durante todo este tiempo.

- A todas aquellas personas que se quedaron en el rincón de la memoria.

- Gracias a todos, son el motor de mi vida.

ÍNDICE

INTRODUCCIÓN

CAPÍTULO 1. ASPECTOS GENERALES DE LOS SISTEMAS OPERATIVOS

CAPÍTULO 2. PROCESOS

CAPÍTULO 3. EL HARDWARE

CAPÍTULO 1. ASPECTOS GENERALES DE LOS SISTEMAS OPERATIVOS

1.1 Historia

La invención de una serie de lenguajes, cada uno más cómodo que sus predecesores, puede continuar indefinidamente hasta llegar a uno adecuado. Cada lenguaje se basa en su predecesor, por lo que podemos pensar en una computadora que emplea esta técnica como una serie de capas o niveles, uno encima del otro. Existe una relación importante entre un lenguaje y una máquina virtual. Cada una tiene cierto lenguaje de máquina, que consiste en todas las instrucciones que la máquina puede ejecutar.

Las primeras computadoras digitales, en los años cuarenta, sólo tenían dos niveles: el ISA (Instruction Set Architecture) y del cual hablaremos con más detalle en los capítulos siguientes, en el que se efectuaba toda la programación; y el de lógica digital, que ejecutaba esos programas. Los circuitos del nivel de lógica digital eran complicados, difíciles de entender y construir, y poco confiables.

En 1949 Maurice Wilkes, investigador de la Universidad de Cambridge, sugirió la idea de diseñar una computadora de tres niveles a fin de simplificar drásticamente el hardware. Esta máquina tendría un intérprete inalterable integrado (el microprograma) cuya función sería ejecutar programas del nivel ISA (Arquitectura del Conjunto de

Instrucciones) por interpretación. Puesto que el hardware ahora sólo tendría que ejecutar microprogramas, que tienen un conjunto limitado de instrucciones; (en lugar de programas del nivel ISA, que tienen un conjunto de instrucciones mucho más grande), se requerirían menos circuitos electrónicos. Dado que en esa época los circuitos se construían con bulbos, tal simplificación prometía reducir el número de bulbos y, por tanto, mejorar la confiabilidad.

Durante los cincuenta se construyeron unas cuantas máquinas de tres niveles. En los años sesenta se construyeron más, y para 1970 la idea de interpretar el nivel ISA con un microprograma, y no directamente con los circuitos, era sobresaliente.

En estos primeros años, la mayor parte de las computadoras eran de "taller abierto", lo que significa que el programador tenía que operar la máquina personalmente. Junto a cada una había una hoja para reservar. Un programador que quería ejecutar un programa reservaba cierto tiempo, digamos de las 5 p.m. a las 8 p.m. Al llegar la hora, el programador se dirigía al cuarto de máquinas con un paquete de tarjetas perforadas de 80 columnas (un medio de entrada primitivo) en una mano y un lápiz en la otra. Al llegar al cuarto de la computadora, al programador que ese momento estuviera utilizando la computadora se le hacía la observación de que su tiempo había terminado y se le conducía amablemente hacia la puerta de salida y el nuevo programador se hacia cargo de la computadora.

Este procedimiento, con variaciones menores, fue normal en muchos centros de cómputo durante años; obligaba a los programadores a aprender a operar la máquina y a saber qué hacer cuando tenía un desperfecto, que era muy seguido. La máquina pasaba mucho tiempo de ocio mientras la gente llevaba tarjetas de un lado a otro del recinto o se rascaba la cabeza tratando de averiguar por qué sus programas no estaban funcionando correctamente.

Alrededor de 1960, la gente trató de reducir el desperdicio de tiempo automatizando la tarea del operador. Un programa llamado *sistema operativo* estaba cargado en la computadora todo el tiempo. El programador incluía ciertas tarjetas de control junto con el programa, y el sistema operativo las leía y ejecutaba sus instrucciones.

Uno de los primeros sistemas operativos de mayor uso fue el FMS (Fortran Monitor System), en la IBM 709. Aunque el sistema operativo se diseñó para automatizar la tarea del operador (de ahí el nombre), también fue el primer paso en la creación de una nueva máquina virtual.

En años subsecuentes, los sistemas operativos se volvieron más y más sofisticados. Se añadieron nuevas instrucciones, recursos y características al nivel ISA hasta que apareció un nuevo nivel. Algunas de las instrucciones de este nuevo nivel eran idénticas a las del nivel ISA, pero otras, sobre todo las de entrada salida, eran totalmente distintas. Las nuevas instrucciones se conocían como "macros" del

sistema operativo o llamadas al supervisor. El término usual ahora es llamadas al sistema.

Los sistemas operativos evolucionaron también en otros sentidos. Los primeros leían paquetes de tarjetas e imprimían sus salidas en una impresora. Esta organización se llamaba *sistema por lotes*. Por lo regular había una espera de varias horas entre el momento en que se presentaba un programa y el momento en que los resultados estaban listos. Desarrollar software era difícil en esas circunstancias.

A principios de los años sesenta investigadores del Dartmouth College. MIT, y otros sitios crearon sistemas operativos que permitían a programadores comunicarse directamente con la computadora. En esos sistemas, terminales remotas se conectaban a la computadora central a través de líneas telefónicas. Muchos usuarios compartían la misma cpu (Unidad Central de Proceso). Un programador podía introducir un programa con el teclado y obtener los resultados impresos casi de inmediato en su oficina, en la cochera de su casa o donde fuera que se instalara la terminal. Esos sistemas se llamaban, y se siguen llamando, sistemas de tiempo compartido.

1.2 ¿Qué es un sistema operativo?

Antes de continuar, tenemos que decir que un sistema operativo puede tener varias definiciones dependiendo del punto de vista que tomemos, veamos porque. Una computadora sin software, es incapaz de efectuar muchas tareas comunes sin tener algún apoyo. Tareas

como presentar información en la pantalla o la transferencia de datos a disco necesitaban una serie de instrucciones complejas.

Hace algunos años cada programa era responsable del control de todas las tareas y, en consecuencia, la escritura de los programas era una labor mucho más complicada de lo que es en la actualidad. Por ejemplo, para guardar información en memoria, algunas computadoras necesitaban saber en qué lugar de la memoria se debía colocar. De igual manera, la comunicación entre el usuario y la computadora era mucho más tediosa; con frecuencia los usuarios introducían los comandos accionando interruptores, y recibían las respuestas mediante filas de luces.

En los años sesenta, un sistema operativo se podría haber definido como el *software que controla al hardware*. Sin embargo, existe una tendencia significativa a la transferencia de las funciones del software al *firmware*, es decir, al *microcódigo*. Dicha tendencia se ha pronunciado tanto que es probable que en algunos sistemas, las funciones codificadas en firmware sobrepasen pronto aquellas codificadas en software.

Se puede imaginar un sistema operativo como los programas, instalados en el software o el 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 de la computadora: los procesadores, los medios de almacenamiento, los dispositivos de entrada/salida, los dispositivos de comunicación y los datos. Los sistemas operativos realizan muchas funciones, como proporcionar la interfaz de usuario, permitir que los usuarios compartan entre sí el hardware y los datos, evitar que los usuarios se interfieran mutuamente, planificar la distribución de los recursos entre los usuarios, facilitar la entrada y salida, recuperarse de los errores, contabilizar el uso de los recursos, facilitar las operaciones en paralelo, organizar los datos para lograr un acceso rápido y seguro, y manejar las comunicaciones en red.

En la actualidad, la utilización de computadoras es relativamente fácil, en parte, debido a las mejoras de los sistemas operativos. El sistema operativo maneja las operaciones comunes, permitiendo al usuario y a los programas concentrarse en las cuestiones menos técnicas; razón por la cual el usuario adquirió su computadora. Casi ningún usuario de computadoras piensa en el sistema operativo, ni siquiera cuando esta usándolo. La gente utiliza la computadora para realizar alguna tarea práctica, por lo general mediante un *programa de aplicación*, como un procesador texto o algún programa de contabilidad.

Como sea que se decida definir a los sistemas operativos, lo importante es no olvidar que constituyen una parte integral del ambiente de cómputo y, por tanto, necesitan ser comprendidos en alguna medida por todos los usuarios de computadoras.

1.3 Funciones de los sistemas operativos

Las funciones clásicas del sistema operativo se pueden agrupar en las tres categorías siguientes:

- Gestión de los recursos de la computadora.
- Ejecución de servicios para los programas.
- Ejecución de los mandatos de los usuarios.

Como se muestra en la figura 1.0; el sistema está formado conceptualmente por tres capas principales.

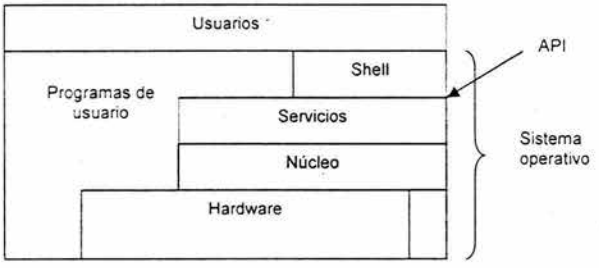


Figura 1.0 Niveles del sistema operativo.[1]

La capa más cercana al hardware se denomina núcleo (*kernel*) y es la que gestiona los recursos hardware y suministra la funcionalidad

[1] Carretero Peréz, Jesús. De Miguel Anasagasti, Pedro. **Sistemas Operativos una Visión Aplicada.** McGraw-Hill. Madrid, España. 2002. 721 páginas.

básica del sistema operativo.El núcleo es la parte del sistema operativo en la que se hace verdaderamente el trabajo. Es la base fundamental, es el que se encarga de la comunicación entre hardware y software, así como de la administración del mismo.

La capa de *servicios o llamadas al sistema* ofrece a los programas servicios en forma de una interfaz de programación o API (Application Programming Interface). Desde el punto de vista de los programas, esta capa extiende la funcionalidad de la computadora, por lo que suele decirse que el sistema operativo ofrece una máquina virtual extendida a los programas. De esta forma se facilita la elaboración de éstos, puesto que se apoyan en las funciones que le suministra el sistema operativo.

Las llamadas al sistema son unas instrucciones especiales que utiliza el sistema para su comunicación con los programas y de estos hacia el sistema. Están estrechamente relacionadas con las funciones de la librería estándar en cada sistema operativo, ya que para cada una de estas existe su similar como llamada al sistema.

La capa de *intérprete de mandatos o shell* suministra una interfaz a través de la cual el usuario puede dialogar de forma interactiva con la computadora. El shell recibe los mandatos u órdenes del usuario, los interpreta y, si puede, los ejecuta. Se comporta como un bucle infinito que se está repitiendo constantemente.

El diagrama de la figura 1.1 muestra este proceso.

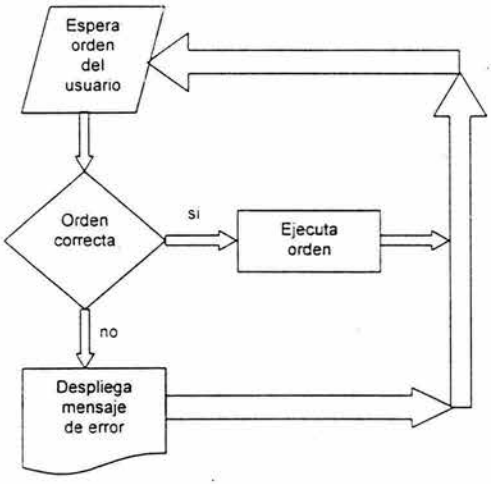


Figura 1.1 Diagrama de flujo que muestra como trabaja el interprete de comandos (shell) del sistema operativo.

- Espera una orden del usuario. En el caso de interfaz textual, el shell está pendiente de lo que escribe el usuario en la línea de mandatos. En las interfaces gráficas está pendiente de los eventos del apuntador (ratón) que manipula el usuario, además, de los del teclado.

- Analiza la orden y, en caso de ser correcta, la ejecuta, para lo cual emplea los servicios del sistema operativo.

- Concluida la orden vuelve a la espera.

1.4 Tipos de sistemas operativos

Los sistemas operativos también son susceptibles de clasificarlos o dividirlos dependiendo de sus características:

- Por la estructura de su kernel
- Por los servicios ofrecidos
- Por el soporte de los servicios

1.4.1 Sistema operativo por la estructura de su kernel

Un sistema operativo es un programa grande y complejo que está compuesto por una serie de componentes con funciones bien definidas. Cada sistema operativo estructura estos componentes de distinta forma. En función de esta estructura se pueden agrupar los sistemas operativos en dos grandes grupos: sistemas operativos monolíticos y sistemas operativos estructurados.

1.4.1.1 Monolíticos

Es la estructura de los primeros sistemas operativos. Un sistema operativo de este tipo no tiene una estructura clara y bien definida. Todos sus componentes se encuentran integrados en un único

programa (el sistema operativo) que se ejecuta en un único espacio de direcciones. En este tipo de sistemas todas las funciones se ejecutan en modo núcleo.

Estos sistemas operativos han surgido, normalmente, de sistemas sencillos y pequeños a los que se les ha ido añadiendo un número mayor de funciones. Esto les ha hecho evolucionar y crecer hasta convertirlos en programas grandes y complejos, formados por muchas funciones situadas en un mismo nivel. El problema que plantea este tipo de sistemas radica en lo complicado que es modificar el sistema operativo para añadir nuevas funciones y servicios. En efecto, añadir una nueva característica al sistema operativo implica la modificación de un gran programa, compuesto por miles de líneas de código fuente y funciones, cada una de las cuales puede invocar a otras cuando así lo requiera.

Este tipo de sistemas esta 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 enlazador (linker).

- Buena definición de parámetros de enlace entre las distintas rutinas existentes, lo 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, entre otros.

- Generalmente están hechos 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.

1.4.1.2 Sistemas por capas

A medida que fueron creciendo las necesidades de los usuarios, se fueron perfeccionando los sistemas. Se hizo necesaria una mayor organización del software, del sistema operativo, donde una parte del sistema contenía subpartes y esto se organizó en forma de niveles. El sistema operativo se dividió en pequeñas partes, de tal forma que cada una de ellas estuviera perfectamente definida del resto.

Se constituyó una estructura jerárquica o de niveles. El primero de los cuales fue denominado THE (Technische Hogeschool Eindhoven), desarrollado por Edsger W. Dijkstra, físico holandés y teórico de la Universidad de Leiden. Se puede pensar también en estos sistemas como si fueran multicapa; Multics y Unix (sus derivados) caen en esa categoría.

Los sistemas operativos con este tipo de enfoque están diseñados para utilizar hardware más avanzado. Si se cuenta con el

apoyo apropiado del hardware, los sistemas operativos se pueden dividir en fragmentos más pequeños y adecuados. Así, el sistema operativo puede mantener un control mucho mayor sobre la computadora y las aplicaciones que lo usan. Los desarrolladores tienen más libertad para modificar el funcionamiento interno del sistema.

En un sistema por capas, el sistema operativo se organiza como una jerarquía de capas, donde cada una ofrece una interfaz clara y bien definida a la capa superior y solamente utiliza los servicios que le ofrece la capa inferior.

La principal ventaja que ofrece este tipo de estructuras es la modularidad y ocultación de la información. Una capa no necesita conocer como se ha diseñado la capa sobre la que se construye, únicamente necesita conocer la interfaz que ofrece. Esto facilita enormemente la depuración y verificación del sistema, puesto que las capas se pueden ir construyendo y depurando por separado. Podemos depurar la primera capa sin preocuparnos por el resto del sistema porque, por definición, sólo utiliza el hardware básico para realizar sus funciones. Una vez depurada la primera capa, se puede dar por sentado su funcionamiento correcto mientras se trabaja con la segunda capa, y así sucesivamente.

Cada capa está construida utilizando sólo operaciones provistas por capas del nivel inferior. Una capa no necesita saber cómo se han diseñado dichas operaciones; sólo necesita saber qué hacen. De este

modo, cada capa oculta la existencia de ciertas estructuras de datos, operaciones y hardware, de la capas de niveles más altos.

Otra forma de ver este tipo de sistema es la denominada de anillos concéntricos. En el sistema de anillos, cada uno tiene una apertura, conocida como puerta o trampa (trap), por donde pueden entrar las llamadas de las capas inferiores. De esta forma, las zonas más internas del sistema operativo o núcleo del sistema estarán más protegidas de accesos indeseados desde las capas más externas. Las capas más internas serán, por tanto, más privilegiadas que las externas.

El principal problema del enfoque de capas tiene que ver con la definición apropiada de las distintas capas. Puesto que una capa sólo puede usar las capas que están en un nivel más bajo, la planificación debe ser muy cuidadosa. Por ejemplo, el *driver* del dispositivo para el almacenamiento auxiliar (espacio de disco utilizado por los algoritmos de memoria virtual) debe estar en un nivel más bajo que el de las rutinas de gestión de memoria, porque esta última función necesita poder usar el almacenamiento auxiliar.

Otro problema del diseño por capas es que tienden a ser menos eficientes que otros tipos. Por ejemplo, cuando un programa de usuario quiere ejecutar una operación de E/S (entrada/salida), ejecuta una llamada al sistema que transfiere el control por una trampa a la capa de E/S, lo cual invoca la capa de gestión de memoria, pasa después a la capa de planificación del cpu, y por último al hardware. En cada capa,

los parámetros podrían modificarse, podría haber necesidad de pasar datos o parámetros. Cada capa implica un gasto extra por el procesamiento de la llamada al sistema, y el resultado neto es una llamada que tarda más que en un sistema sin capas.

1.4.1.3 Máquinas virtuales

Conceptualmente, el sistema de una computadora está compuesto por capas. El hardware es el nivel más bajo. El núcleo que se ejecuta en el siguiente nivel utiliza las instrucciones de hardware para crear un conjunto de llamadas al sistema que las capas exteriores pueden usar. Así, los programas del sistema que están arriba del núcleo pueden usar llamadas al sistema o instrucciones de hardware, y en cierto sentido estos programas no distinguen entre las dos. Por tanto, aunque se accede de forma diferente a las llamadas y a las instrucciones, ambas proporcionan funcionalidad que el programa puede aprovechar para crear funciones todavía más avanzadas. Los programas del sistema, a su vez, tratan al hardware y a las llamadas al sistema como si estuvieran en el mismo nivel.

Algunos sistemas llevan este esquema un paso más allá permitiendo a los programas de aplicación invocar fácilmente los programas del sistema. Aquí también, aunque los programas del sistema están en un nivel más alto que las demás rutinas, los programas de aplicación podrían ver todo lo que está debajo de ellos en la jerarquía como si formara parte de la máquina misma. Este enfoque

de capas se lleva a su conclusión lógica en el concepto de maquina virtual.

Utilizando planificación del cpu y técnicas de memoria virtual, un sistema operativo puede crear la ilusión de que múltiples procesos se ejecutan cada uno en su propio procesador con su propia memoria (virtual). Desde luego, lo normal es que el proceso tenga características adicionales, como llamadas al sistema y un sistema de archivos, que el hardware desnudo no provee. El enfoque de máquina virtual, en cambio, no proporciona funcionalidad adicional, sino que presenta un interfaz que es idéntica al hardware desnudo subyacente. Cada proceso recibe una copia (virtual) de la computadora subyacente (Figura 1.2).

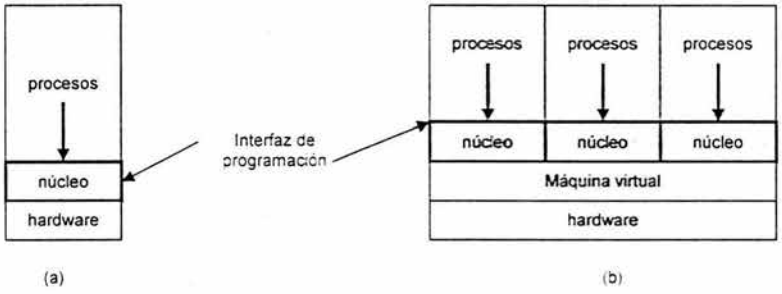


Figura 1.2 Modelos de sistemas. (a) Máquina no virtual. (b) Máquina virtual.

Los recursos físicos de la computadora se comparten para crear las máquinas virtuales. Se puede usar planificación del cpu para

compartir el cpu y crear la ilusión de que los usuarios tienen su propio procesador.

Así, cada usuario recibe su propia máquina virtual, y puede ejecutar cualquiera de los sistemas operativos y paquetes de software que estén disponibles en la máquina subyacente. El software de máquina virtual se ocupa de multiprogramar varias máquinas virtuales dentro de una máquina física, pero no necesita considerar software de apoyo a los usuarios.

Ahora, desde otro punto de vista, simplemente se trata de un tipo de sistemas operativos que presentan una interfase 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
- 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 maquinas 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.

1.4.1.4 Modelo Microkernel

El tipo más reciente de sistemas operativos es el denominado cliente/servidor o microkernel, que puede ser ejecutado en la mayoría de las computadoras. Este sistema sirve para toda clase de aplicaciones, por tanto, es de propósito general y cumple con las mismas actividades que los sistemas operativos convencionales.

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 E/S (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, E/S, 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.

La figura 1.3 presenta la estructura de un sistema operativo con estructura cliente/servidor. Como puede apreciarse, el sistema operativo está formado por diversas partes, cada una de las cuales puede desarrollarse por separado.

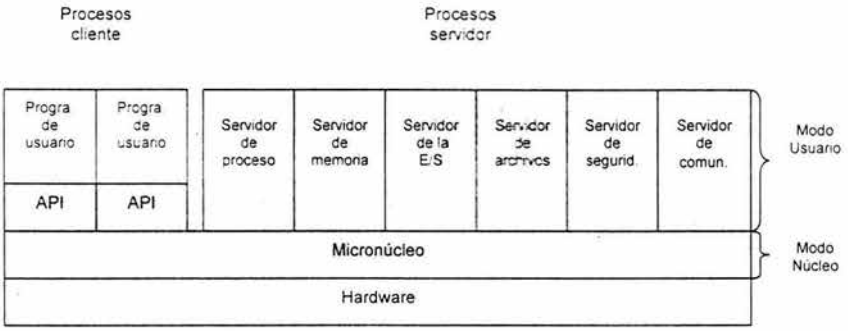


Figura 1.3 Estructura cliente-servidor en un sistema operativo.[2]

No hay una definición clara de las funciones que debe llevar a cabo un micronúcleo. La mayoría incluyen la gestión de interrupciones, gestión básica de procesos, de memoria y servicios básicos de comunicación entre procesos.

El punto de vista usual es el de establecer la mayoría de las funciones del sistema operativo en los procesos de usuario. Para solicitar un servicio en este tipo de sistemas, como por ejemplo crear un

[2] Idem.

proceso, el proceso de usuario (proceso denominado cliente) solicita el servicio al servidor del sistema operativo correspondiente, en este caso al servidor de procesos. A su vez, el proceso servidor puede requerir los servicios de otros servidores, como es el caso del servidor de memoria. En este caso, el servidor de procesos se convierte en cliente del servidor de memoria.

La ventaja de este modelo es la gran flexibilidad que presenta. Cada proceso servidor sólo se ocupa de una funcionalidad concreta, lo que hace que cada parte pueda ser pequeña y manejable. Esto a su vez facilita el desarrollo y depuración de cada uno de los procesos servidores.

En cuanto a las desventajas, es importante citar que estos sistemas presentan una mayor sobrecarga en el tratamiento de los servicios que los sistemas monolíticos. Esto se debe a que los distintos componentes de un sistema operativo de este tipo se ejecutan en espacios de direcciones distintos, lo que hace que su activación requiera más tiempo.

1.4.2 Sistemas operativos por los servicios ofrecidos

Esta clasificación es la más usada y conocida desde el punto de vista del usuario final.

1.4.2.1 Monousuario

Los sistemas operativos monousuario son aquéllos que soportan a 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.

Como ejemplo de este tipo se pueden mencionar a:

- Ms DOS (Microsoft Disk Operating System)
- Windows 9x y Me
- Mac OS (antes de OS X)

1.4.2.2 Multiusuario

Los sistemas operativos multiusuario son capaces de dar servicio a más de un usuario a la vez, ya sea por medio de varias terminales conectadas a la computadora o por medio de sesiones remotas en una red de comunicaciones. No importa el número de procesadores en la máquina ni el número de procesos que cada usuario puede ejecutar simultáneamente.

Algunos sistemas tipo Unix como GNU/Linux ofrecen además de la posibilidad de trabajar con terminales remotas; el trabajo con terminales, esta característica permite emular en una sola máquina el

trabajo con varias terminales remotas ya que es posible iniciar sesión como diferentes usuarios (o como el mismo varias veces) utilizando el mismo hardware pero de forma tal que parezca que cada uno trabaja con una terminal independiente.

Ejemplos de sistemas operativos multiusuario:

- Unix (y sus derivados)
- Windows 2000
- Mac OS X

1.4.2.3 Monotarea

Los sistemas operativos del tipo monotarea son aquellos que sólo permiten una tarea a la vez por usuario. Puede darse el caso de un sistema multiusuario y monotarea al mismo tiempo, en el cual se admiten varios usuarios al mismo tiempo pero cada uno de ellos puede estar haciendo sólo una tarea a la vez.

Como ejemplos podemos mencionar a:

- MS-DOS
- Windows 3.X, 95 (estos sistemas tan sólo simulaban la multitarea, pero realmente pertenecen a esta clasificación)

1.4.2.4 Multitarea

Un sistema operativo multitarea es aquél que le permite al usuario estar realizando varias labores al mismo tiempo. Por ejemplo, podemos estar escuchando música con un reproductor de cd, escribiendo un programa, copiando archivos entre directorios y descargando música de Internet. Además, es común encontrar en ellos interfaces gráficas orientadas al uso de menús y el ratón, lo cual permite un rápido intercambio entre las tareas para el usuario, mejorando su productividad.

Como ejemplos podemos citar a:

- Mac OS
- Unix (y derivados)
- Windows 98, Me, 2000 y XP.

1.4.2.5 Monoproceso

Un sistema operativo monoprocesador es aquél que es capaz de manejar solamente un procesador, de manera que si la computadora tuviera más de uno le sería inútil. El ejemplo más típico de este tipo de sistemas es el MS-DOS y MacOS en sus versiones antiguas.

1.4.2.6 Multiproceso

Un sistema operativo multiprocesador se refiere al número de procesadores, que obviamente es más de uno, y que es capaz de usarlos para distribuir su carga de trabajo.

Generalmente estos sistemas trabajan de dos formas:

- Simétricamente
- Asimétricamente

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 (hilos o threads) son enviados indistintamente a cualquiera de los procesadores disponibles teniendo, teóricamente, una mejor distribución y equilibrio en la carga de trabajo. Se dice que un hilo es la parte activa en memoria 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.

1.4.3 Sistemas operativos por la forma de ofrecer sus servicios

Esta clasificación también se refiere a una visión externa, que en este caso se refiere a la del usuario: el cómo accede a los servicios. Bajo esta clasificación se pueden detectar dos tipos principales; sistemas operativos de red y sistemas operativos distribuidos.

1.4.3.1 Sistemas operativos de red

Los sistemas operativos de red se definen como aquellos que tiene la capacidad de interactuar con sistemas operativos en otras máquinas por medio de un medio de transmisión con el objeto de intercambiar información, transferir archivos, ejecutar comandos remotos y un sin fin de otras actividades.

El punto crucial de estos sistemas es que el usuario debe conocer la ubicación de los recursos a los que desee acceder (por ejemplo la dirección y/o nombre de la máquina remota, ruta o camino al recurso compartido), e incluso en ocasiones puede ser necesario incluso conocer la sintaxis de una serie de instrucciones necesarias para la utilización del recurso (vale la pena aclarar que esto cada vez es menos frecuente debido al surgimiento de interfaces más amigables con el usuario, aunque sin lograr superar la flexibilidad que una línea de comandos puede brindar).

1.4.3.2 Sistemas operativos distribuidos

Los sistemas operativos distribuidos abarcan los servicios de red, logrando integrar recursos (impresoras, unidades de respaldo, memoria, procesos, unidades centrales de proceso) en una sola máquina virtual a la que el usuario accede en forma transparente. Es decir, ahora el usuario ya no necesita saber la ubicación de los recursos, sino que los conoce por nombre y simplemente los usa como si todos ellos fueran locales a su lugar de trabajo habitual.

La labor del sistema operativo distribuido es la de permitir a las aplicaciones hacer uso de los diversos procesadores, memorias, discos y en general todo dispositivo conectado al sistema múltiple, tal como si se encontraran físicamente en la misma máquina. Existen dos razones principales para crear o adoptar sistemas distribuidos: por necesidad (debido a que los problemas a resolver son inherentemente distribuidos) o porque se desea tener más confiabilidad y disponibilidad de recursos.

El procesamiento en paralelo es fundamental en los grandes centros de investigación como por ejemplo el Earth Simulator Center (Centro de Simulación de la tierra) laboratorio de investigación en aspectos relacionados con nuestro planeta y el ser humano, el cual trabaja gracias a un súper cluster que se compone de 40 clusters con alrededor de 16 máquinas cada uno, integrados bajo un sistema operativo basado en Unix denominado SUPER-UX (una versión mejorada del Unix de NEC); esta supercomputadora aparece en la primera posición de la lista del top 500 de sitios de supercomputo

acompañado en la tercera posición de un cluster de máquinas GNU/Linux en el Laboratorio Nacional Lawrence Livermore del Departamento de Energía de los Estados Unidos.

1.5 Comparación de sistemas operativos

El sistema operativo es el componente de cualquier sistema de cómputo que administra los recursos del sistema, además de que, provee la base sobre la cual operan los diversos servicios del usuario, como son utilerías para la administración del sistema, así como también aplicaciones más complejas.

Es un hecho que la mezcla más importante de sistemas operativos es la de las computadoras personales, la mayoría de arquitectura Intel o compatibles, que trabajan básicamente bajo la plataforma Windows en cualquiera de sus versiones; y en las empresas el sistema central puede ser mas variado, pero con una mayor tendencia hacia el sistema Unix.

No podemos negar la existencia que para ciertas aplicaciones algunos sistemas operativos brindan un mejor rendimiento al usuario, como puede ser la arquitectura Motorola (Apple Macintosh), el cual es muy usado en aplicaciones de diseño gráfico.

Hacer una comparación de sistemas operativos es un proceso que no es sencillo, debido a la gran variedad de estos, además de que es delicado, ya que se deben tener en cuenta varios aspectos. Se

podría hablar de sistemas operativos de algunas compañías, también se puede mencionar a los sistemas operativos de tipo Unix ó podemos referirnos a los sistemas operativos que están orientados al diseño gráfico y manejo de imágenes y video. El camino que se utilizará para esta comparación es como se describe a continuación.

Nuestra comparación se va a dividir en dos partes, nos vamos a referir a los sistemas operativos desde un punto de vista del administrador de sistemas y desde un punto de vista de ingeniería.

El primero de los puntos, el del administrador de sistemas, se refiere a la parte que el administrador debe mostrar al gerente de la empresa, a un jefe de división o incluso a un usuario final, ya que tiene que ver con el rumbo de la empresa o de algún proyecto. En este punto vamos a considerar cuatro aspectos que son fundamentales y que involucran el punto de vista de todos.

En el punto de vista de ingeniería, existe una gran cantidad de factores que tomar en cuenta en una evaluación de este tipo, pero a pesar de esto, consideraremos solo tres aspectos fundamentales:

Tabla 1.0 Aspectos a tomar en cuenta en la comparación de sistemas operativos

Administrador de Sistemas	Ingeniero en computación
El esquema de licencias y precios	La compatibilidad con otras plataformas

La estabilidad y desempeño	La portabilidad
La facilidad de uso	Los requerimientos de hardware
El soporte	

Una vez definido el esquema a seguir, es necesario mencionar que sistemas operativos vamos a evaluar:

- FreeBDS
- NetBSD
- OpenBSD
- GNU/Linux
- Mac OS X
- Windows 98
- Windows 2000

Como podemos ver, nos estamos refiriendo a las tres plataformas que mencionamos al inicio Unix, Apple Macintosh y Microsoft, con lo cual se trata de ser lo más imparcial posible.

Antes de empezar a hacer la comparativa es necesario dar una breve descripción de cada sistema operativo para saber de quienes estamos hablando.

FreeBSD

FreeBSD es un derivado de BSD Unix, la versión de Unix desarrollada en la Universidad de Berkeley. FreeBSD es desarrollado y mantenido por un gran número de desarrolladores. FreeBSD es ideal para proporcionar servicios de Internet o Intranet. Proporciona servicios de red robustos, incluso en situaciones de alta carga, haciendo un uso eficaz de la memoria para mantener buenos tiempos de respuesta con cientos o miles de procesos simultáneos de usuario.

NetBSD

Es otro de los descendientes de la versión de Unix proporcionada por la Universidad de Berkeley (4.4 BSD Lite y 386 BSD). Es de software libre, lo que quiere decir que se puede conseguir sin costo alguno. Sus metas están bien definidas, es extremadamente portable, sin descuidar la seguridad y estabilidad como la mayoría de los derivados de Unix.

Open BSD

Es un sistema operativo libre de tipo Unix. Es multiplataforma, basado en 4.4 BSD. Esta concentrado en la portabilidad, el

cumplimiento de normas y regulaciones, corrección del código, *seguridad proactiva* y criptografía integrada. Incluye emulación en binarios para la mayoría de los programas de los sistemas SVR4 (Solaris), FreeBSD, Linux, BSD/OS, SunOS y HP-UX.

GNU/Linux

Linux es el kernel de un sistema operativo desarrollado por Linus Benedict Trovalds, está basado en Minix, que es un sistema clónico de Unix desarrollado por Andrew S. Tanenbaum, y por lo tanto, como los anteriores esta basado en Unix. GNU/Linux es un conjunto de herramientas que hace de Linux un sistema operativo. GNU significa GNU is Not Unix y es un proyecto mundial de software libre, iniciado en 1984 por Richard M. Stallman.

Mac OS X

Apple lo describe, como un sistema operativo súpermoderno que proporciona la potencia de Unix más la simplicidad y elegancia de Macintosh. Diseñado para ser los cimientos de la plataforma Macintosh en la próxima década. Mac OS X combina nuevas tecnologías basadas en estándares abiertos, que lo convierten no sólo en el sistema operativo personal más estable, compatible e interactivo del mundo, sino también en el más apasionante e innovador.

Windows 98

Es uno de los sistemas operativos de escritorio con mayor cantidad de usuarios. Marca una pauta importante en Microsoft, ya que por primera vez maneja un sistema de archivos de 32 bits, además de que soporta la multitarea (con algunas reservas).

Windows 2000

La familia de sistemas operativos de Windows 2000 ofrece una gran flexibilidad a los usuarios de negocios crecientes, y es de los sistemas operativos de red multipropósito para cualquier tipo de empresa. Con el uso de su ambiente gráfico su manejo se vuelve muy fácil.

Cabe hacer mención que, se ha omitido a las versiones anteriores de Windows que realizaban la tarea de ser una interfase gráfica del sistema operativo MS-DOS. También a Windows 95 ya que los logros significativos de esta versión fueron hasta el 98.

1.5.1 Desde el punto de vista del administrador de sistemas

Tomando en consideración el punto de vista del usuario final, vamos a estudiar con más detalle cada uno de los aspectos mencionados al inicio de esta comparación.

1.5.1.1 Esquema de licencias y precios

Cuando hablamos de un esquema de licencias hacemos referencia al acuerdo que contraen dos partes: el productor y el usuario; y que por lo general el segundo descuida (la mayoría de las veces se limita a aceptar el acuerdo sin cuando menos leerlo brevemente), pero que constituye una defensa para el primero.

Se encuentra muy ligado al precio, (a pesar de ser dos conceptos diferentes) debido a que por lo general, lo que el usuario compra son *licencias*, los esquemas de licencias son muy diversos y por lo general varían entre cada compañía productora.

FreeBSD

Este sistema operativo es libre y gratuito, su kernel está bajo la licencia BSD (la Universidad de Berkeley da licencia de usar el código) al igual que gran cantidad de sus aplicaciones, en especial aquellas de integración con el sistema o configuración, sin embargo, como la mayoría de los Unix, utiliza algunas de las herramientas generadas por el proyecto GNU, por lo cual, algunas porciones de FreeBSD tienen licencia por la GPL (licencia pública general).

Se puede conseguir de forma gratuita, descargándolo desde Internet, sin embargo, también es posible encontrarlo por algún costo, por lo regular bastante bajo comparado a otros sistemas operativos.

NetBSD

Este sistema operativo, por ser otra variante de BSD y derivado directo de 4.4 BSD y 386 BSD, se encentra liberado bajo la licencia BSD, esto implica, obviamente, que se garantiza su libre redistribución para cualquier fin.

OpenBSD

Al igual que los anteriores, es software libre y se rige por la licencia BSD, la cual garantiza su libre redistribución, aunque sujeto a los siguientes lineamientos generales:

- La redistribución debe mantener cualquier nota de copyright del original, preservando así la propiedad intelectual.

- El software regido por esta licencia se entrega sin ningún tipo de garantía.

GNU/Linux

GNU/Linux se encuentra liberado y protegido a su vez por la licencia pública general o GPL. Esta licencia se destaca por ofrecer las siguientes libertades básicas:

- Libertad para ejecutar el programa, con cualquier propósito.

- Libertad para modificar el programa y adaptarlo a sus necesidades. (Para que esta libertad sea efectiva en la práctica, es necesario disponer del código fuente, porque modificar un programa sin disponer del código fuente es difícil).

- Libertad para redistribuir copias.

- Libertad para distribuir versiones modificadas del programa, de tal manera que la comunidad pueda beneficiarse con sus mejoras.

En cuanto al precio, GNU/Linux se encuentra disponible en Internet sin costo alguno, sin embargo los propios fabricantes de las diferentes distribuciones o cualquier persona o empresa puede cobrar por el sistema; los precios pueden ser tan dispares, que como ejemplo podemos mencionar los siguientes: USD$ 18 por el conjunto de 7 CDs de Debian 3.0, USD$ 69 por el powerpack de Mandrake 9.2 (7 cds, 1 manual), USD$ 200 el ProSuite de Mandrake 9.2 (9 CDs, 1 DVD, 2 manuales, incluyendo también la versión para este sistema operativo de la base de datos comercial IBM DB2 8.1, además de soporte telefónico), o incluso USD$ 750 por el SUSE LINUX Enterprise Server (cabe aclarar que también incluye software propietario y soporte).

Mac OS X

Merece atención especial, ya que este sistema operativo incluye un esquema libre como propietario, es decir, este sistema tiene su base

fundamental (core) en el sistema libre Darwin, el cual está liberado bajo la licencia Apple Public Source License (APSL), la cual en su versión 2.0 ha sido aprobada por la Free Software Foundation (FSF) como una licencia de software libre. Sin embargo, también tiene componentes propietarios, como por ejemplo, la interfaz gráfica Aqua (introducida originalmente con la aparición de los iMac), por lo cual su esquema de licencia también es propietario, el Mac OS X Phanter (la última versión hasta este momento) tiene dos alternativas de compra, 1 usuario (es decir sólo se puede instalar en una computadora) por USD$ 129, y el Family Pack para 5 usuarios por USD$ 199.

Windows 98

El esquema de licencias de la Microsoft ofrece varias alternativas, desde la licencia OEM para las computadoras con este sistema preinstalado, hasta licencias de tipo *campus agreement,* es decir, licencias en masa cuando se requiere poner licencia a un número significativo de máquinas. Sin embargo, ya no es posible de forma oficial conseguir este sistema operativo ya que su productor a decidido (unilateralmente) dejar de venderlo, tan sólo es posible conseguirlo a través de terceros que aún tengan remanentes.

Windows 2000

Pertenece a la familia de sistemas operativos de Microsoft, el esquema de licenciamiento es similar al de Windows 98, aunque para el caso de esta versión es aún un poco más complicado, ya que no sólo

se debe tener en cuenta el número de máquinas que necesitan licencia, sino también el número de procesadores en cada máquina, en especial si pasa de dos, ya que la licencia (y el mismo programa) deben ser versiones especiales.

Windows 2000 es ofrecido en 4 versiones:

- Professional
- Server
- Advanced Server
- Datacenter Server. La licencia de Datacenter sólo permite su adquisición como software preinstalado en una serie limitada de servidores certificados. Las opciones de soporte técnico serán las que hagan de Datacenter un producto atractivo. Sin un contrato de soporte técnico, lo único que se obtendrá es el software acompañado de un costosísimo servidor, sin olvidar los dolores de cabeza que dará el comprobar que se han efectuado correctamente los cambios de configuración del servidor. En cambio, con un contrato de soporte técnico, se tendrá línea directa con el servicio de soporte para sistemas de alta prioridad de Microsoft, así como la garantía de que el sistema funcionará el 99,99 por ciento del tiempo, de que se cambiará el hardware y el software en menos de cuatro horas en caso de que exista algún problema y de que las aplicaciones funcionarán. Es por

estas razones que en esta comparativa no hablaremos más de esta versión de Windows.

Cada una de estas, posee características que la hacen más adecuada a determinadas necesidades, la siguiente tabla resume el propósito general de la versión y su precio.

Tabla 1.1 Las diferentes versiones de Windows 2000

Versión	Descripción	Precio unitario
Professional	Es la versión para equipos de escritorio, provee únicamente funcionalidades básicas.	USD$ 319
Server	Provee funcionalidades de servidor de archivos e impresión, así como capacidades de servidor Web.	USD$ 999
Advanced Server	Además de las capacidades de Windows 2000 Server provee servicios de alta disponibilidad.	USD$ 3999

1.5.1.2 Estabilidad y desempeño

A pesar que existen varias pruebas para medir el rendimiento de un sistema operativo, estas por lo general se desempeñan en condiciones generadas, es decir, por lo general son conducidas en laboratorios y con máquinas especialmente puestas a punto para la

comparación, por lo tanto, no son muy fiables cuando se trata de evidenciar la realidad del uso diario.

Cuando se habla de estabilidad, se hace referencia a las capacidades de la máquina de soportar la carga sin tener alteraciones que impliquen bloqueos o reinicio, lo cual se deriva en tiempo perdido e incluso posible perdida de la información; el desempeño se refiere a la relación trabajo/tiempo, es decir, a la mejor administración de los recursos del sistema logrando una ejecución rápida de las aplicaciones pero sin descuidar la estabilidad.

FreeBSD

Es uno de los sistemas operativos más estables del mercado, obviamente esto sucede en gran parte por ser una variante de Unix, aunque va un poco más allá. En cuanto a rendimiento también podemos observar que una buena cantidad de sitios en Internet se encuentran soportados por este sistema operativo.

NetBSD

El desarrollo de NetBSD se caracteriza por la limpieza en la codificación para permitir la portabilidad, sin embargo, esto trae otro beneficio añadido como lo es, la estabilidad y el desempeño, la principal meta de NetBSD no es brindar nuevas características que puedan resultar incompatibles entre algunas plataformas, sino que por el

contrario procura escribir bien y mantener un código sin mayores problemas para que su mantenimiento sea fácil.

OpenBSD

Aunque el principal propósito de OpenBSD es la seguridad, esto no implica que su rendimiento y estabilidad no sean los más adecuados, esto se debe a que por más seguro que pueda llegar a ser un sistema operativo si no es lo suficientemente estable como para que esa seguridad extra sea utilizable, este no llegaría a ser ni medianamente utilizado.

GNU/Linux

Aunque el rendimiento puede variar entre una distribución y otra, además de que depende en gran medida del hardware sobre el cual se está ejecutando, GNU/Linux es un sistema operativo bastante confiable tanto en rendimiento como en estabilidad.

Debido a su gran flexibilidad es posible personalizarlo de múltiples maneras, de tal forma que cumpla con los requerimientos necesarios, es decir, si necesitamos una estación de trabajo con una buena interfaz gráfica para el hogar o la oficina, podríamos no arrancar los servidores u otras aplicaciones y dejar simplemente lo que necesitamos, si por el contrario, si lo que se desea es un servidor para el Web o algún otro tipo de servicio como FTP (File Transport Protocol) o bases de datos, no es necesario que se tenga una interfaz gráfica, ya

que va a consumir recursos de memoria y procesador que pueden en algún momento necesitar los servidores.

En cuanto a estabilidad como buen derivado de Unix, GNU/Linux es altamente estable.

Mac OS X

La estabilidad en los sistemas Macintosh ha sido altamente calificada a través de su historia, ahora, teniendo como base otro gran sistema operativo en términos de estabilidad. Apple ha logrado una muy buena combinación y este nuevo producto se considera ahora suficiente competencia, incluso a sistemas operativos como GNU/Linux o FreeBSD en términos de confiabilidad. Así mismo el trabajo a nivel de rendimiento en esta plataforma parece superar a las versiones anteriores de este mismo sistema operativo.

Windows 98

La estabilidad es precisamente una de las grandes carencias de este sistema operativo, casi cualquier usuario por muy poco tiempo que halla pasado frente a una máquina operada por este sistema, habrá encontrado algún bloqueo del sistema, reinicio, o la tristemente famosa "pantalla azul" indicándole que ha ocurrido algún tipo de fallo y que muy posiblemente sólo se pueda solucionar reiniciando el sistema completo.

Claro que es de reconocer el adelanto que significó respecto a su antecesor (Windows 95) tanto en estabilidad como en desempeño, este último en especial al incorporar por primera vez la multitarea de forma real.

Windows 2000

Este es uno de los mejores sistemas Windows que han sido desarrollados, integra el rendimiento y estabilidad de NT (frente a Windows 95 y 98), con la interfaz unificada y más amigable que hizo de Windows 95 una novedad. Su estabilidad especialmente en las versiones Server y Advanced Server puede llegar a compararse con la de algunos Unix, hecho que le ha permitido incluso ganar una porción del mercado de los servidores para Internet.

1.5.1.3 Facilidad de uso

Lo que para algunos usuarios puede ser de gran facilidad (como una línea de comandos) para otros puede resultar absolutamente incomprensible. Por esta razón en este apartado tan sólo se comentaran algunas alternativas de las interfaces tanto de línea de comando como de interfaz gráfica.

FreeBSD, NetBSD, OpenBSD, GNU/Linux

Como todo Unix, en estos sistemas la interfaz primaria es la línea o interprete de comandos, la cual les provee gran flexibilidad, pero esta

puede llegar a ser intimidante para algunos usuarios, en especial si están acostumbrados a trabajar exclusivamente con interfaces gráficas como ocurre con otros sistemas operativos; sin embargo esto no es ningún impedimento para que los BSD y GNU/Linux puedan tener interfaces gráficas bastante amigables y que le permitan al usuario realizar algunas de las operaciones a las que tiene acceso por el interprete de comandos.

Todo Unix tiene como motor gráfico el sistema X-Window, y específicamente estos cuatro sistemas operativos utilizan una implementación libre de este servidor, a la que se denomina XFree 86, por esta razón la mayoría de gestores de ventanas e incluso los dos metaproyectos de entornos de escritorio (GNOME y KDE) pueden ser utilizados en estas plataformas.

La facilidad de uso es muy relativa, pero por lo menos en estos sistemas operativos es posible contar con varias alternativas que se ajusten a las necesidades particulares de cada usuario, por lo cual esta facilidad se puede garantizar.

Mac OS X

Desde sus inicios los sistemas Macintosh se han destacado por su facilidad de uso y por su amigable interfaz gráfica, y esta última versión (Mac OS X Panther v10.3) obviamente no es una excepción, su sistema de ventanas Aqua se integra de forma nativa con el sistema de ventanas XFree86.

Windows 98 y Windows 2000

A diferencia de los Unix, la interfaz primaria para Windows es la GUI (interfaz gráfica de usuario) y no la línea de comandos, mientras que en Unix la mayoría de las cosas que se hacen a través del interprete de comandos es posible (aunque algunas veces de forma más complicada) hacerlas mediante la GUI, en estos sistemas por el contrario es bastante complicado (y muchas veces imposible) desempeñar la mayoría de acciones en la GUI a través de una interfaz de línea de comandos.

Para la mayoría de usuarios, la interfaz más sencilla para trabajar y visualmente más atractiva es esta (aunque en mayor medida esta apreciación se debe a la falta de conocimiento de otras alternativas), la organización de la interfaz en los sistemas Windows sumado a su amplia difusión, hacen que para el usuario sea un poco más rápido encontrar lo que necesita, ya que por tratarse de un producto bajo el control de una sola compañía, siempre se va a encontrar la misma organización de los menús en todos los sistemas.

1.5.1.4 Soporte

El soporte se refiere a la ayuda que otra compañía o persona pueda prestar en caso de tener algún problema. Existen diversos tipos de soporte, siendo los más comunes el telefónico y aquel en el cual el técnico se traslada hasta donde se encuentra la máquina afectada. Sin embargo, no siempre es necesario que el soporte se preste de esta

forma, en muchas ocasiones también se puede conseguir soporte a través de Internet, o bien, buscando la documentación que resuelva nuestro problema o incluso consultando en foros o grupos de interés, en los cuales otras personas pueden ayudar con posibles soluciones concretas.

Para el caso de esta evaluación, nuevamente se han agrupado algunos sistemas debido a que también comparten las mismas características en cuanto a soporte técnico.

FreeBSD, NetBSD, OpenBSD

Debido a que estos sistemas operativos no están desarrollados por alguna compañía que dicte sus lineamientos, el soporte de estos productos es algo opcional en el sentido de que no hay *personal certificado* por la empresa desarrolladora para llevar a cabo esta tarea, esto aparentemente significaría que no es posible encontrar soporte técnico calificado para estos sistemas.

Sin embargo, esto dista mucho de la realidad, ya que en todo el mundo existen cientos de empresas que se dedican a este trabajo, en los sitios Web de cada uno de estos sistemas los desarrolladores recomiendan algunas de ellas, aunque siempre es posible encontrar otras con igual o mayor calidad en su trabajo.

La ventaja de ser sistemas libres permite que cualquiera pueda estudiar su funcionamiento hasta en los aspectos más profundos (cosa

que no suele ocurrir con los soportes de otros sistemas donde el código es cerrado) y por tanto ofrecer en cualquier momento la ayuda más adecuada, incluso en algunas ocasiones esta mismas empresas que brindan soporte pueden tener entre sus integrantes ha personas que han participado directamente en el desarrollo y por lo tanto están lo suficientemente calificadas como cualquier otra, e incluso mucho más.

Pero esta no es la única forma de encontrar ayuda para la solución de problemas con estos sistemas operativos, en Internet se encuentra gran documentación sobre como hacer las cosas, muy buenos tutoriales y en general muchas ideas que le pueden servir considerablemente; así mismo también siempre es posible contactar con alguno de los muchos grupos de interés y pedir ayuda cuando se encuentren dificultades, por lo regular en poco tiempo otros usuarios que ya conocen la solución para ese problema responderán con la solución, e incluso con varias alternativas de solución; esto conlleva que a su vez, quién en algún momento dado haya recibido soporte pueda en algún momento auxiliarle a alguien más que lo necesite, conformando así vínculos de comunidad colaborativa.

GNU/Linux

GNU/Linux es un caso parecido a lo anteriormente escrito, pero es necesario resaltar que la mayoría de distribuciones que son desarrolladas por una empresa, poseen la opción de adquirirlas con soporte técnico por lo general vía telefónica o por correo electrónico, lo

cual para algunas instituciones es de carácter fundamental a la hora de decidirse por una solución de software.

Mac OS X

Apple sólo ofrece soporte a través de Internet a sus usuarios registrados, aunque también mantiene gran cantidad de documentación para que pueda ser accedida libremente, así mismo a través de está página también es posible encontrar información acerca de grupos de usuarios en todo el mundo. Además, por ser fundamentalmente un derivado Unix hay muchas cosas que se hacen de forma similar, por lo cual gran parte de la abundante documentación para sistemas operativos de este tipo también es de utilidad para el Mac OS X.

Windows 98 y Windows 2000

En esta familia de plataformas aunque existen muchísimas empresas que de forma independiente brindan soporte, también existen algunas que son contratadas o certificadas directamente por el fabricante, además como esta empresa tiene representación en varios países del mundo, también es posible establecer contacto por vía telefónica.

Para terminar con el punto de vista del administrador de sistemas, veamos la siguiente tabla que muestra un resumen de todo lo descrito anteriormente.

Tabla 1.2 Resumen desde el punto de vista del administrador de sistemas

Sistema	Licencia	Precio	Estabilid. /Desemp.	Facilidad de uso	Soporte
Free BSD	BSD y GPL	Gratuito	Excelente	Línea de comandos	Internet
Net BSD	BSD	Gratuito	Bueno	Línea de comandos	Internet
Open BSD	BSD	Gratuito	Regular	Línea de comandos	Internet
GNU/Linux	GPL	Es gratuito pero hay distribuciones que tienen un costo	Excelente	Línea de comandos	En Internet, vía telefónica ó con la compañía encargada de la distribución
Mac OS X	APSL	El core es gratuito pero tiene alguna aplicaciones que tienen un costo	Bueno	Interfaz gráfica	Da soporte a por Internet a sus usuarios registrados
Win 98	OEM y Campus Agreement	Ya no es vendido por la compañía	Malo	Interfaz gráfica	Vía telefónica, con empresas

					certificadas o directamente con Microsoft
Win 2000	OEM y Campus Agreement	Depende de la versión y del número de procesadores	Bueno	Interfaz gráfica	Vía telefónica, con empresas certificadas o directamente con Microsoft

1.5.2 Desde el punto de vista de la ingeniería

En la sección anterior se describieron algunas características que son de mayor interés para un usuario final común y corriente; en esta sección vamos a ir un poco más allá para tratar de crear una perspectiva más completa de cada sistema y de esta manera ver cual de ellos cumple todas nuestras necesidades de la mejor manera.

1.5.2.1 Compatibilidad con otras plataformas

Cuando se habla de compatibilidad con otras plataformas se deben tener en cuenta diferentes conceptos:

- Soporte a sistemas de archivos diferentes al nativo: este concepto se asocia a la capacidad que tiene una plataforma de leer y/o escribir en sistemas de archivos de plataformas diferentes.

- Ejecución de aplicaciones compiladas para otras plataformas: algunos sistemas operativos son capaces de ejecutar aplicaciones que fueron originalmente diseñadas para trabajar en una plataforma diferente, la mayoría de las veces (en especial cuando son sistemas bastante diferentes) esto se lleva a cabo por intermedio de otra aplicación que actúa como una capa intermedia entre el sistema operativo y la aplicación no nativa, esta capa permite emular las llamadas al sistema y algunos otros procesos del sistema operativo para que el programa crea que se está comunicando con la plataforma para la cual fue compilado.

- Compartir recursos en red con otras plataformas: no se puede abandonar la idea que el mundo actual es un mundo "conectado", es decir, las redes y la comunicación entre diferentes máquinas es algo cada vez más indispensable, por lo tanto, el sistema operativo debe ser capaz de comunicarse con sistemas diferentes en otras máquinas, y obviamente brindarles la posibilidad de utilizar sus recursos (por lo general archivos e impresoras, aunque realmente podría ser cualquier dispositivo).

FreeBSD, NetBSD, OpenBSD

Estos sistema operativos poseen soporte para lectura y escritura de una buena variedad de sistemas de archivos, entre los que cabe destacar: FAT 16 (DOS, todos los Windows antes de Windows 98), FAT32 (Windows 98, Windows Me), ext2 y ext3 (GNU/Linux); para sólo lectura se pueden reseñar: NTFS (Windows NT, 2000 entre otros).

Estos sistemas operativos, poseen compatibilidad binaria para ejecutar aplicaciones de GNU/Linux, BSD y SCO; también es posible ejecutar aplicaciones DOS (a través del emulador DOSEmu) y Windows (con Wine).

Para poder compartir recursos con otros Unix, se utiliza el protocolo NFS (Network File System), mientras que para compartir con Windows (protocolo SMB) se utiliza Samba.

GNU/Linux

GNU/Linux posee soporte entre otros para los siguientes tipos de sistemas de archivos:

- JFS: el Journaled File System es un sistema de archivos desarrollado por IBM para sus servidores de alto nivel (enterprice) y que comparte con la comunidad en su decidido apoyo a este sistema operativo.

- XFS: este sistema de archivos es el utilizado por SGI en su sistema operativo y que ha puesto a disposición de la comunidad.

- Minix FS: fue el primero que soportó Linux, como su nombre lo indica es el sistema de archivos del Minix del profesor Tanenbaum.

- FAT: Linux soporta tanto FAT 16 como FAT 32.

- NTFS: el sistema de archivos de NT (además de Windows 2000 y Windows XP) es totalmente soportado para la lectura; para la escritura las nuevas versiones del kernel (2.6.x) ya traen esta opción.

- ADFS: es el que utiliza el sistema operativo RISC OS, bastante usado en procesadores RISC. Hay que resaltar que aún se encuentra en etapa experimental.

- BeFS: el File System del sistema operativo BeOS es posible leerlo aunque también se encuentra en etapa experimental.

- BFS: es el Unixware Boot Filesystem.

- EFS: es el antiguo sistema de archivos del sistema operativo Irix de SGI.

- HPFS: sistema de archivos del OS/2.

En cuanto a ejecución de aplicaciones de otras plataformas, Linux soporta el formato binario ELF, el cual es utilizado por otros sistemas Unix, por lo cual es posible correr algunas aplicaciones creadas para estos otros sistemas operativos, además en GNU/Linux existen los emuladores DOSEmu y Wine, por lo que es posible ejecutar aplicaciones diseñadas para sistemas Windows.

La compatibilidad en red con otras plataformas se da por medio de los protocolos NFS, SMB (samba), NCP (NetWare Core Protocol) para compartir con Novel NetWare, Coda (un avanzado sistema de archivos para redes utilizado en varios Unix), InterMezzo (sistema de archivos bastante utilizado en sistemas distribuidos de alta disponibilidad), Andrew File System (otro sistema de archivos distribuido).

Mac OS X

Como Darwin está basado en tecnología FreeBSD y Mach, tiene la posibilidad de soportar los formatos binarios de otros Unix, en especial los del tipo BSD y Linux, pero como además tiene componentes Mac puede soportar también aplicaciones hechas para versiones anteriores de ese sistema, en cuanto a ejecución de aplicaciones Windows, el Mac OS X, lo puede hacer por medio de un programa llamado Virtual PC, el cual debe comprarse de manera independiente. Para compartir recursos utiliza Samba y NFS.

Windows 98

Windows 98 soporta aplicaciones para DOS y Windows (95/98, con versiones anteriores algunas veces tiene inconvenientes), también es posible por medio de software de terceros ejecutar binarios para Mac. A nivel de sistemas de archivos únicamente soporta FAT 16 y FAT 32.

Utiliza de forma nativa el protocolo SMB (NetBIOS) para compartir archivos e impresoras con otros Windows, es posible instalar software de terceros para trabajar en redes Novell Netware.

Windows 2000

Soporta aplicaciones para Windows de 32 bits (Windows 98 en adelante), las aplicaciones de 16 bits (DOS, Windows 3.x, 95) por lo general no se pueden ejecutar.

Los sistemas de archivos soportados son FAT 32 y NTFS. a forma de compartir archivos también es por SMB, aunque por medio de un programa vendido por la misma compañía (Services For Unix, SFU) es posible compartir por medio de NFS, al igual que Windows 98 también se puede compartir con Novell Netware.

La tabla que aparece a continuación muestra un resumen de la compatibilidad de los sistemas operativos con otras plataformas tomado en cuenta el sistema de archivos, las aplicaciones que puede ejecutar y como comparten recursos en red.

Tabla 1.3 Compatibilidad con otras plataformas

Sistema operativo	Sistema de archivos	Ejecución de aplicaciones	Compartir recursos en red
Free BSD Open BSD Net BSD	FAT 16, FAT 32, ext2, ext3 y NTFS	Compatibilidad binaria para ejecutar aplicaciones de GNU/Linux. BDS, SCO, DOSEmu y Wine	Protocolo NFS y SMB
GNU/Linux	JFS, XFS, Minix FS, FAT 16, FAT 32, NTFS, ADFS, BeFS, BFS, EFS, HPFS	Soporta el formato binario ELF, DOSEmu y Wine	Protocolo NTFS, SMB, NCP, CODA, InterMezzo y Andrew File System
Mac OS X	Ext2, ext3, FAT 16, FAT 32, NTFS	Utiliza Virtual PC para aplicaciones de Windows. Soporta los formatos binarios de Unix	Protocola SNB y NFS
Win 98	FAT 16 y FAT 32	Mediante software de terceros es posible ejecutar archivos binarios para Mac	Protocolo SMB (NetBios). Es posible instalar software de terceros para trabajar en redes

			Novell
Win 2000	FAT 16, FAT 32, NTFS	Soporta aplicaciones de 32 bits	Protocolo SMB, utiliza SFU (Service For Unix), NFS y comparte archivos con Novell.

1.5.2.2 Portabilidad

Cuando en sistemas operativos se habla de portabilidad, esto hace referencia a la posibilidad de utilizarlos en diferentes tipos de procesadores incluso si tienen arquitecturas o diseños diferentes.

FreeBSD, OpenBSD y GNU/Linux

Además de la tradicional plataforma Intel x86 (386, 486, Pentium, Pentium II, Pentium III, Pentium IV), estos sistemas operativos también se pueden ejecutar en un buen número más de plataformas, entre ellas se puede resaltar: Intel ia64, Sparc, UltraSparc, PPC (Macintosh), Alpha y Alpha64.

Cada día es mayor la cantidad de dispositivos portátiles (relojes, PDA´s, equipos de cocina, equipos de entretenimiento, XBOX, Sega, PlayStation, entre otros) en los que se puede encontrar como sistema operativo una versión reducida de Linux.

MacOS X

Este sistema operativo está diseñado para la arquitectura típica de Mac, es decir los procesadores Power Mac o Power PC (PPC).

NetBSD

En términos de portabilidad este sistema es el vencedor absoluto, como ya se mencionó, esta es precisamente su meta principal, encontrándose portado en más de 50 diferentes procesadores que van desde los grandes servidores (Sparc, Alpha), pasando por los de escritorio (x86, PPC), e incluso en dispositivos portátiles como consolas de juegos y otros.

Windows (cualquiera de ellos)

Los sistemas Windows, cualquiera del que estemos hablando, en un principio solamente se ejecutaban sobre arquitecturas Intel x86.

Sin embargo, existe una versión modificada denominada Windows CE, orientada al mercado móvil, además las versiones para servidores cuentan con soporte para ser instalados en plataformas Alpha, Motorota 680, Mips 400, HP-PA de Hewlett Packard, los Sparc Risc de Sun Microsystems, el RS/6000 de IBM y támbien se esta desarrollando para algunas versiones del PowerPc de Apple.

En la siguiente tabla mostramos que procesadores pueden ser utilizados con cada uno de los sistemas operativos.

Tabla 1.4 Portabilidad de los sistemas operativos

Sistema operativo	Portabilidad
Free BSD Open BSD GNU/Lninux	Intel x86 (386, Pentium, Pentium II, Pentium III, Pentium IV), Intel ia64, Sparc, UltrSparc, PPC (Macintosh), Alpha, Alpha 64, PDA´s, XBOX, Sega y Play Station
Mac OS X	Power Mac ó Power PC, PPC
Net BSD	Es utilizado en más de 50 procesadores
Windows (cualquiera de ellos)	Intel x86, Alpha, Motorola 680, Mips 400, HP-PA, Sparc Risc, RS/6000 y algunos Power PC

1.5.2.3 Requerimientos mínimos de hardware

Los sistemas FreeBSD, GNU/Linux, NetBSD y OpenBSD pueden correr en diferentes plataformas de hardware, por lo cual es lógico que sus requerimientos pueden variar, por este motivo, hemos decidido sólo mostrar en la siguiente tabla, los requerimientos de la plataforma x86, excepto en el caso del Mac OS X, donde se utilizará el procesador Power PC.

Tabla 1.5 Requerimientos mínimos de hardware

Sistema operativo	Procesador mínimo	Mínimo de RAM	Espacio mínimo en disco duro
FreeBSD	386	16 Mb para el programa de instalación, después de esto se puede reducir hasta 4 Mb	10Mb a 50Mb
GNU/Linux	386	4 Mb	100 Mb
Mac OS X	PowerPC G3	128 Mb	2 Gb
NetBSD	386	4 Mb	50 Mb
OpenBSD	386	8 Mb	50 Mb
Win 98	486	16 Mb	165 Mb
Win 2000 Professional	Pentium 133 Mhz	64 Mb	650 Mb
Win 2000 Server / Advanced S	Pentium 133 Mhz	128 Mb	1 Gb
Win XP Profesional	Pentium 233 Mhz	128 Mb	1.5 Gb

Es bueno aclarar que estos requerimientos deben variar bastante dependiendo de las necesidades de la instalación, es decir no se necesitaran los mismos requerimientos (en especial en el caso de los

Unix) si se desea tener sólo una máquina que actué como router o como servidor de archivos, para lo cual no es necesario una interfaz gráfica, que si se deseara tenerla para el uso cotidiano (suite de oficina, juegos, entre otros).

Incluso, estos requerimientos mínimos pueden llegar a cambiar, existen proyectos (ya funcionando), que se configuran con capacidades menores, por ejemplo, núcleos de sistemas como BSD y Linux, con los cuales se puede tener una computadora ejecutándose con el sistema operativo en un disquete tradicional. Así mismo como ya se dijo si se desea una interfaz gráfica los requerimientos de hardware pueden llegar hasta el punto de duplicarse (o algo más). Esto también aplica para los sistemas Windows, excepto por que en estos no es posible suprimir la interfaz gráfica ni algunos otros componentes.

CAPÍTULO 2. PROCESOS

Durante la fase de diseño de un sistema operativo se deciden las funcionalidades básicas y características del sistema: el sistema de archivos a utilizar, el módulo de gestión de memoria, la gestión de los procesos, entre otras cosas. En este apartado vamos a analizar, solamente, las decisiones más internas de la gestión de procesos, ya que estas nos servirán de base para el desarrollo de nuestro prototipo, y que son resueltas de diferente forma según el sistema operativo de que se esté hablando.

Los primeros sistemas de cómputo sólo permitían la ejecución de un programa a la vez. Este programa, asumía el control total del sistema y tenía acceso a todos sus recursos. Los sistemas de cómputo actuales, permiten cargar múltiples programas en la memoria y ejecutarlos de forma concurrente. Esta evolución requirió de un control más firme, lo cual nos lleva a compartir más los recursos entre los distintos programas concurrentes. Esto dio pie al concepto de proceso.

La multiprogramación (multitarea) es la multiplexción de los recursos del sistema entre una serie de programas activos. Los beneficios potenciales de la ejecución concurrente de programas, incluyen el aumento del rendimiento de la utilización de los recursos y la velocidad de respuesta de un sistema. Se puede decir que los procesos son un mecanismo para definir y gestionar la ejecución concurrente de los programas bajo el control de un sistema operativo.

2.1 Concepto de proceso

Informalmente, un proceso es un programa en ejecución. Un proceso, es una instancia de un programa en ejecución. Un proceso es más que el *código del programa*: también incluye la actividad actual, representada por el valor del *contador de programa* y el contenido de los *registros* del procesador. Generalmente, un proceso también incluye la *pila* (*stack*) del proceso, que contiene datos temporales (como los parámetros de subrutinas, direcciones de retorno y variables temporales), y una *sección de datos*, que contiene variables globales. La ejecución de un proceso debe proceder de manera secuencial.

Hay que recalcar que, un programa por sí solo no es un proceso; un programa es una entidad pasiva, como el contenido de un archivo guardado en disco; mientras que un proceso es una entidad activa, como el contador de programa especificando la siguiente instrucción que se ejecutará, y un conjunto de recursos asociados.

2.2 Modelo de cinco estados

Todo proceso dependiendo de la forma como se ha diseñado el sistema operativo puede tener dos o más estados, un estado, como su nombre lo indica, es una condición especial de la ejecución del proceso en un instante determinado de tiempo. El modelo más sencillo es en donde sólo existen dos estados (figura 2.0):

- En ejecución

- No ejecución

Esto quiere decir que, el proceso puede en un momento estarse ejecutando y algún tiempo después no.

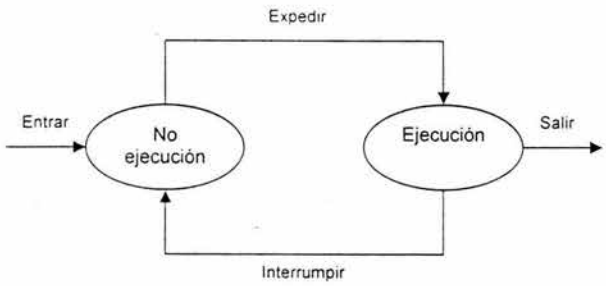


Figura 2.0 El modelo más sencillo es de dos estados.[3]

El primer estado siempre es "No ejecución", ya que este se da tan pronto se crea el proceso y queda en espera para su próxima ejecución. En cualquier momento, cuando el proceso se encuentra en estado "En ejecución" puede ocurrir un suceso interno o externo que lo lleve de nuevo al estado "No ejecución", es de hacer notar que, este modelo no permite el uso de interrupciones, debido a que no hay un estado de "bloqueo" o similar que permita detener por momentos la ejecución del proceso y, reanudarla después desde el punto en donde fue interrumpida.

[3] Stallings, William. **Sistemas Operativos.** 2a. Edición. Prentice Hall. Madrid, 1997. 709 páginas

Este modelo, como podemos observar, es bastante sencillo, pero sería demasiado limitado para satisfacer las necesidades de un sistema operativo moderno, por lo cual se han ideado modelos con mayor cantidad de estados; uno de los más completos otorga la posibilidad al proceso de tener cinco estados; y sobre esta base vamos a trabajar.

A continuación se muestra un diagrama de este modelo:

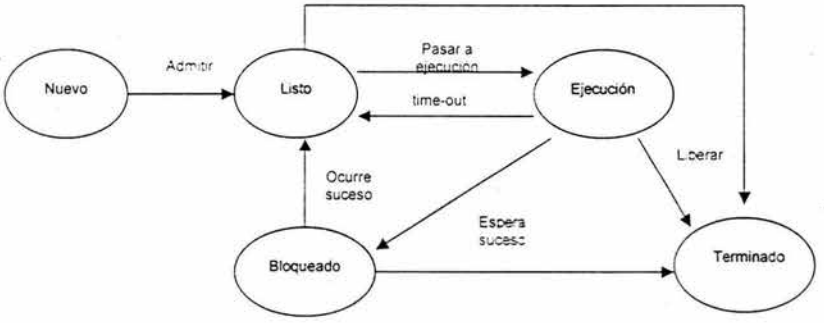


Figura 2.1 Modelo de procesos de cinco estados.[4]

Los cinco estados de este diagrama son los siguientes:

- Ejecución: se refiere al proceso que está actualmente en ejecución.

[4] Idem

- Listo: es aquel en donde, el proceso está preparado para ejecutarse en cuanto se le dé la oportunidad.

- Bloqueados: se refiere al proceso que no puede ejecutarse hasta que se produzca cierto suceso, como la terminación de una operación de E/S.

- Nuevo: es cuando un proceso se acaba de crear, pero que aún no ha sido admitido por el sistema operativo en el grupo de procesos ejecutables.

- Terminado: un proceso que ha sido excluido por el sistema operativo del grupo de procesos ejecutables, ya sea porque se detuvo o porque fue abandonado por alguna razón.

Los estados Nuevo y Terminado son construcciones útiles e importantes para la gestión de procesos, por lo tanto, vamos a desmenuzarlas en el transcurso del capítulo. El sistema operativo puede definir un proceso nuevo en dos pasos.

Primero, se llevan a cabo algunas tareas necesarias de gestión interna. Se le asocia un identificador al proceso y se construyen y asignan algunas tablas necesarias para gestionar el proceso. En este punto, el proceso estará en el estado Nuevo. Esto significa que se han llevado a cabo las tareas necesarias para crear el proceso, pero no se ha asignado aún para su ejecución.

Asimismo, un proceso sale en dos pasos. Primero, el proceso termina cuando llega al punto normal de terminación, cuando se abandona debido a un error irrecuperable o cuando otro proceso con la debida autoridad hace que el proceso abandone.

La terminación pone al proceso en el estado Terminado. En este punto, el proceso ya no se elige más para la ejecución. Sin embargo, las tablas de procesos (estructuras de datos) y otra información asociada con el trabajo son conservadas temporalmente; esto da tiempo a otros programas auxiliares o de soporte para extraer la información necesaria.

A continuación se muestran cada una de las posibilidades por turnos:

- Nulo → Nuevo: se crea un proceso nuevo para ejecutar un programa.

- Nuevo → Listo: el sistema operativo pasará un proceso del estado Nuevo al estado Listo cuando esté preparado para aceptar un proceso más.

- Listo → Ejecución: cuando es hora de seleccionar un proceso nuevo para ejecutar, el sistema operativo elige a uno de los procesos del estado Listo.

- Ejecución → Terminado: el proceso que se está ejecutando es finalizado por el sistema operativo si indica que terminó o si se abandona.

- Ejecución → Listo: la razón más común de esta transición es que el proceso que está en ejecución ha alcanzado el tiempo máximo permitido de ejecución ininterrumpida.

- Ejecución → Bloqueado: un proceso se pone en el estado Bloqueado si solicita algo por lo que debe esperar. como una acción de E/S.

- Bloqueado → Listo: un proceso que está en el estado Bloqueado pasará al estado Listo cuando se produzca el suceso que estaba esperando.

- Listo → Terminado: un proceso padre termina al proceso hijo en cualquier momento. Además, si el padre termina, todos los procesos hijos asociados a él pueden ser finalizados.

- Bloqueado → Terminado: se aplica lo mismo que en el caso anterior.

La figura 2.2 muestra la manera como debe desarrollarse una disciplina de colas (estructuras de datos) para los estados Listo y Bloqueados.

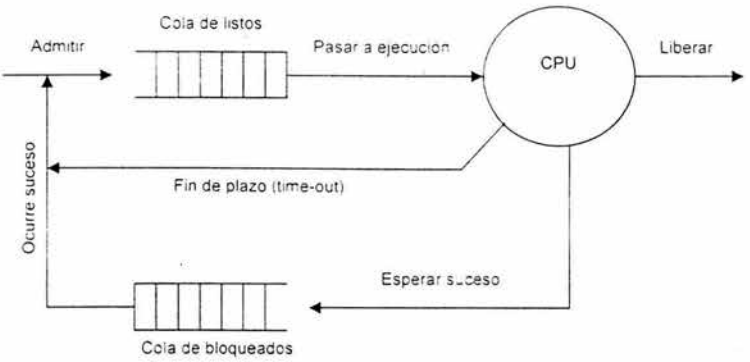


Figura 2.2 Modelo de Colas.[5]

A medida que se admiten procesos en el sistema, se sitúan en la cola de Listos. Cuando llega la hora de que el sistema operativo elija (aunque en realidad el que escoge es el planificador) otro proceso para ejecutar, selecciona uno de la cola de Listos.

Cuando un proceso que se está ejecutando es apartado de la ejecución pueden ocurrir varias cosas: se le puede finalizar, también se puede poner en la cola de Listos, ó en la cola de Bloqueados dependiendo de las circunstancias que se presenten.

Por último, cuando se produce un suceso, todos los procesos de la cola de Bloqueados que están esperando a dicho suceso se pasan a la cola de Listos.

[5] Idem

Esta medida significa que, cuando se produce un suceso, el sistema operativo debe recorrer toda la cola de Bloqueados, buscando aquellos procesos que esperan al suceso.

2.3 Descripción de procesos

Se puede ver al sistema operativo como un administrador de procesos y recursos, ya que es él quien planifica y expide a los procesos para ser ejecutados por el procesador, el que asigna los recursos a los procesos y quien responde a las solicitudes de servicios hechas por los usuarios.

2.3.1 Estructuras de control

Como administrador de procesos y de recursos, el sistema operativo, debe tener información sobre el estado actual de cada proceso y de cada recurso. Existe un método universal para esto y es el que se describe a continuación: el sistema crea y mantiene tablas de información sobre cada entidad que esté administrando.

En la figura siguiente se muestra la idea general de este método, en la cual se muestran cuatro diferentes tipos de tablas: de memoria, de E/S, de archivos y de procesos.

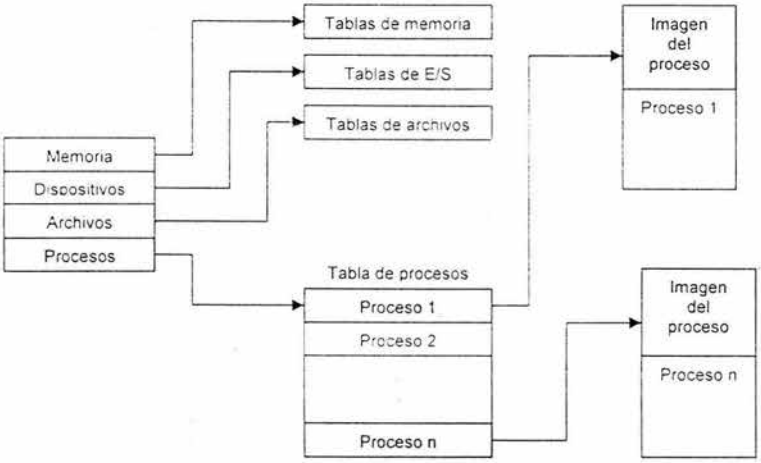


Figura 2.3 Estructura general de las tablas de control del sistema operativo.[6]

Las tablas de memoria del sistema operativo se utilizan para saber la situación de la memoria principal y secundaria (virtual). Estas deben incluir lo siguiente:

- La asignación de memoria principal a los procesos.
- La asignación de memoria secundaria a los procesos.
- El atributo de protección a los segmentos de memoria principal o virtual; es decir, que procesos pueden acceder a ciertos segmentos de memoria compartida.

[6] Idem

- La información necesaria para gestionar la memoria virtual.

Las tablas de E/S. son utilizadas para administrar los dispositivos de E/S del sistema operativo. Si hay una acción que realizar, el sistema operativo debe conocer el estado de la operación y la posición de la memoria principal que se está utilizando para tal acción, así como el origen y destino de la transferencia.

Hay una tabla principal de procesos. con una entrada para cada proceso. Cada entrada contiene, al menos, un apuntador a la imagen de un proceso. Si la imagen del proceso contiene varios bloques, entonces esta información estará guardada directamente en la tabla principal o con referencias cruzadas (estar enlazadas) a entradas de las tablas de memoria.

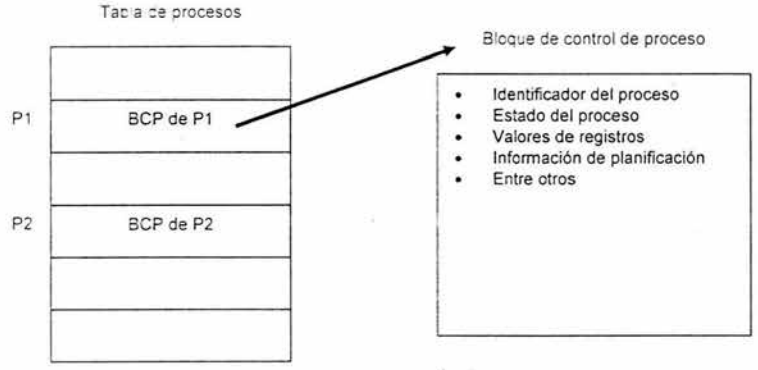


Figura 2.3.1 Muestra un ejemplo de la estructura de una tabla de procesos junto con el BCP correspondiente al proceso uno.

Las tablas de archivos, proporcionan información acerca de la existencia de archivos, su posición en memoria secundaria (disco), su estado actual y otros atributos.

Antes de continuar, debemos señalar que: estas tablas tienen que mantener referencias cruzadas (estar enlazadas) entre ellas. Por ejemplo: la E/S, la memoria y los archivos son administrados en nombre de los procesos. Los archivos que son referidos en las tablas de archivos son accesibles a través de un dispositivo de E/S, y algunas veces estarán en la memoria principal o en la virtual. Así pues, hacen falta referencias cruzadas.

2.4 Estructuras de control de procesos

Si vemos al sistema operativo como un administrador de procesos, entonces se tienen que considerar dos cosas:

- Debe saber dónde está ubicado el proceso y;
- debe conocer los atributos del proceso.

2.4.1 Ubicación de los procesos (Imagen del proceso)

Físicamente, un proceso consta de un *conjunto de programas* que deberán ser ejecutados. Asociados a esto, existen variables locales, globales y constantes definidas. Así pues, un proceso constará,

al menos, de la memoria suficiente para albergar los programas y sus datos. En la ejecución de un programa entra en juego una *pila (stack)*, que se utiliza para llevar la cuenta de las llamadas a procedimientos y de los parámetros que son pasados a los procedimientos.

Por último, asociado a cada proceso, hay una serie de atributos utilizados por el sistema operativo para el control del proceso. Estos se conocen como el *bloque de control del proceso*. A este conjunto de programas, pila y atributos se le llama *imagen del proceso*[7]. Esta, es guardada como un bloque contiguo de memoria. Este bloque se mantiene en memoria secundaria (disco). Para que el sistema operativo pueda administrar el proceso; al menos una parte de su imagen, que contiene la información necesaria a usar por el sistema operativo, debe mantenerse en memoria principal. Para ejecutar el proceso, la imagen completa debe cargarse en la memoria principal.

Por lo tanto, el sistema operativo debe conocer la ubicación de los procesos en el disco y también en la memoria principal. Este esquema permite al sistema operativo tener que traer sólo una parte de un proceso en particular. De este modo, una parte de la imagen de un proceso puede estar en la memoria principal y el resto en memoria secundaria. Por lo tanto, las tablas de procesos deben mostrar la ubicación de cada segmento y/ó página de cada imagen de proceso.

[7] La ubicación de la imagen de un proceso depende del esquema de memoria que sea utilizado.

En la tabla que a continuación se presenta, se muestran los elementos de la imagen de procesos así como una descripción de cada elemento.

Tabla 2.1 Elementos de la imagen de proceso.[8]

Elemento	Descripción
Datos del usuario	La parte modificable del espacio de usuario. Puede guardar datos del programa, una zona para una pila del usuario y programas que pueden modificarse.
Programa de usuario	El programa a ejecutar.
Pila del sistema	Cada proceso tiene una o más pilas asociadas a él. Se utiliza para almacenar los parámetros y las direcciones de retorno.
Bloque de control de proceso	Información necesaria para que el sistema operativo controle al proceso.

2.4.2 Atributos de un proceso

Esta información reside en el *bloque de control del proceso (BCP)*. Esta información se agrupa en tres categorías principales:

- Identificación del proceso.

[8] Stallings, William. **Sistemas Operativos.** 2a. Edición. Prentice Hall, Madrid, 1997. 709 páginas.

- Información del estado del procesador.

- Información de control del proceso.

Con respecto a la *identificación del proceso*: se asigna a cada proceso un número de identificación único. Este identificador es útil en varios sentidos, por ejemplo, muchas de las tablas controladas por el sistema operativo pueden utilizar identificadores de procesos como referencias cruzadas a las tablas de procesos. Cuando los procesos se comunican unos con otros, se utiliza el identificador de proceso para informar al sistema operativo del destino de cada comunicación en particular. O por ejemplo, cuando los procesos crean otros procesos, se utilizan identificadores para señalar al padre y a los descendientes de cada proceso.

Tabla 2.2 Elementos de un bloque de control de procesos.[9]

Identificación de Proceso.
Identificadores.
Los identificadores numéricos que se pueden guardar en el bloque de control de proceso incluyen:
• Identificador de este proceso.
• Identificador del proceso que creó a este proceso (proceso padre).
• Identificador del usuario.

[9] Idem

Información de Estado del Proceso.
Registros Visibles para el Usuario.
Un registro visible para el usuario es aquél al que puede hacerse referencia por medio del lenguaje máquina que ejecuta el procesador.
Registros de Control y de Estado.
Hay varios registros del procesador que se emplean para controlar su funcionamiento:
• *Contador de programa*: contiene la dirección de la próxima instrucción a ser tratada.
• *Códigos de condición*: muestran el resultado de la operación aritmética o lógica más reciente (signo, cero, acarreo, igualdad, desbordamiento).
• *Información de estado*: incluye los indicadores de habilitación de interrupciones y el modo de ejecución.
Punteros de Pila.
Cada proceso contiene una o más pilas LIFO del sistema asociadas. Las pilas se utilizan para almacenar los parámetros y las direcciones de retorno de los procedimientos y de las llamadas al sistema.
Información de Control del Proceso.
Información de Planificación y de Estado.
Ésta es la información que necesita el sistema operativo para llevar a cabo sus funciones de planificación.
• *Estado del proceso*: define la disposición del proceso para ser planificado para ejecutar (en ejecución, listo, bloqueado, etc.).
• *Prioridad*: se puede usar uno o más campos para describir la prioridad de planificación de los procesos.

• *Información de planificación*: ésta dependerá del algoritmo de planificación utilizado.
• *Suceso*: la identidad del suceso que el proceso está esperando antes de poder reanudarse.
Estructuración de Datos.
Se puede dar el caso en el cual un proceso esté enlazado por otros procesos por una cola o alguna otra estructura. Por ejemplo todos los procesos que están en estado de espera de un nivel determinado de prioridad pueden estar enlazados en una cola. El bloque de control de proceso contiene apuntadores a otros procesos para dar soporte a estas estructuras.
Comunicación entre Procesos.
Debe haber varios indicadores, señales y mensajes asociados con la comunicación entre procesos.
Privilegios de los Procesos.
A los procesos se les otorgan privilegios en términos de la memoria a la que pueden acceder y el tipo de instrucciones que pueden ejecutar.
Gestión de Memoria.
Se incluyen punteros a las tablas y/o segmentos que describen la memoria virtual asignada al proceso.
Propiedad de los Recursos y Utilización.
Un proceso puede ser controlador de recursos tales como archivos abiertos, utilización del procesador, etc., esta información es necesaria para el planificador.

La información del estado del procesador; básicamente está formada por el contenido de los registros del procesador. Mientras un

proceso esta ejecutándose, la información está en los registros. Cuando se interrumpe el proceso, toda la información debe guardarse de forma que pueda restaurarse cuando el proceso reanude su ejecución. La naturaleza y el número de registros involucrados dependen del diseño del procesador. Normalmente, en el conjunto de registros se incluyen los registros visibles para el usuario, los registros de control y de estado y los punteros de pila.

Los *registros visibles para el usuario* son aquellos accesibles para los programas de usuario y que se usan para almacenar datos temporalmente. La mayoría de los procesadores incluyen de 8 a 32 registros. Algunas arquitecturas como RISC (Reduced-Instruction Set Computer) disponen de más de 100 registros.

Se emplean varios registros de *control y de estado* para controlar la operación del procesador. La mayoría de éstos, en la mayoría de las máquinas, no son visibles para los usuarios. Algunos de ellos pueden ser visibles para las instrucciones de máquina ejecutadas en modo de control o del sistema operativo. Un registro de control que se encuentra en todos los procesadores es el contador de programa o registro de instrucción, que contiene la dirección de la próxima instrucción que debe leerse. Además, todos los diseños de procesadores contienen un registro o conjunto de registros, a menudo conocido como palabra de estado del programa (PSW, *Program Status Word*) que tiene la información de estado, pero de esto hablaremos con más detalle en el siguiente capítulo.

Por último, uno o más *registros de pila* proporcionan los apuntadores a las pilas empleadas por el sistema operativo para controlar la ejecución de los programas y llevar la cuenta de las interrupciones. La *información de control del proceso*, es la información adicional necesaria para que el sistema operativo controle los diferentes procesos activos.

La siguiente figura muestra la estructura de la imagen de un proceso en memoria virtual.

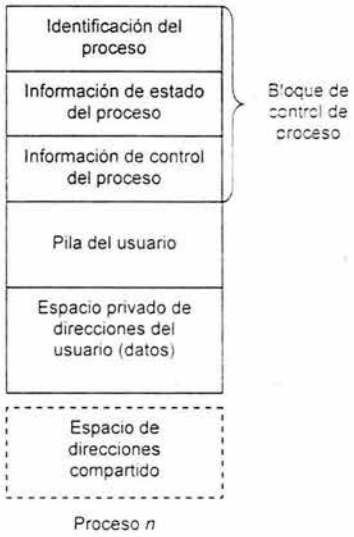


Figura 2.4 Proceso de usuario en memoria virtual.[10]

Esta formada por un bloque de control de proceso, una pila de usuario, el espacio de direcciones privadas del proceso y un espacio de

[10] Idem

direcciones que pueda compartir con otros procesos. La imagen del proceso aparece como un conjunto contiguo de direcciones. Pero en una aplicación real esto depende del esquema de memoria que se este manejando y de la manera en que el sistema operativo maneje las estructuras de control.

Si esto que acabamos de describir lo pudiéramos ver físicamente, entonces, se vería de la siguiente manera:

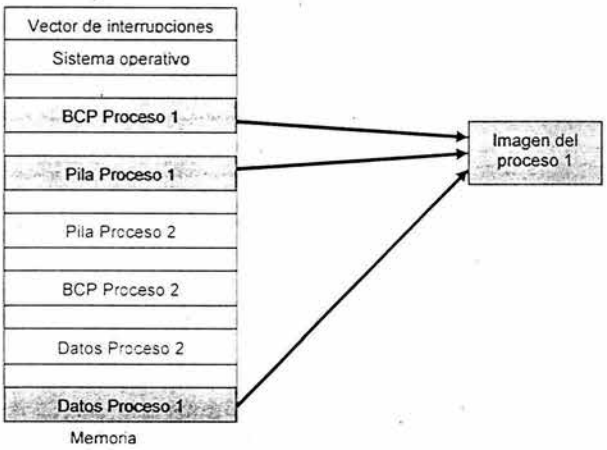


Figura 2.4.1 La figura muestra un mapa de memoria, en el cual se muestran dos procesos, además de la representación de la imagen de uno de ellos.

Como se indica en la tabla 2.2, el bloque de control de proceso contiene información de estructuración, incluyendo apuntadores que permiten enlazar los bloques de control de proceso. Por lo tanto, las

colas que se describen se deben diseñar, como listas enlazadas de los bloques de control de proceso.

Si lo que acabamos de explicar lo pudiéramos ver en forma de esquema entonces, la disciplina de colas de la figura 2.2 debería ser diseñada de la siguiente manera:

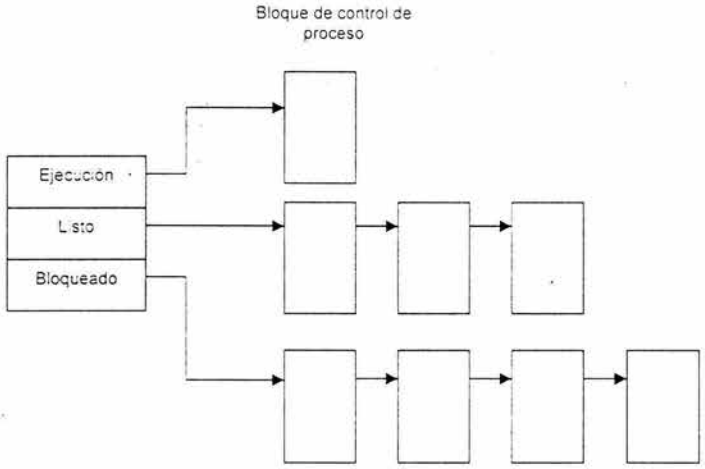


Figura 2.5 Estructura de colas de proceso.[11]

2.5 Bloque de control del proceso

El bloque de control de procesos es la estructura más importante de un sistema operativo. Cada bloque de control de proceso contiene toda la información necesaria de un proceso que el sistema operativo

[11] Idem

necesita. Los bloques son leídos y/o modificados por casi todos los módulos del sistema operativo, incluyendo aquellos que tienen que ver con la planificación, la asignación de recursos, el tratamiento de interrupciones, el análisis y supervisión del rendimiento.

Ahora, una serie de rutinas necesitarán acceder a la información de los bloques de control de procesos. El acceso a esta información no es difícil. Como ya vimos, cada proceso está dotado de un único identificador numérico que se utiliza como índice en una tabla de apuntadores a los bloques de control de procesos. La dificultad no está en el acceso, sino más bien en la protección de la información; y con esto se presentan dos problemas:

- Un error en una sola rutina, como la de tratamiento de interrupciones, puede dañar los bloques de control de procesos, lo que destruiría la capacidad para administrar los procesos afectados.

- Un cambio de diseño en la estructura o en la semántica del bloque de control de procesos podría afectar a varios módulos del sistema operativo.

Esto se puede prevenir haciendo que todas las rutinas del sistema operativo pasen a través de una rutina de manejo, cuya única tarea será la de proteger los bloques de control de proceso y que además se convierta en la única manera que se tenga para poder leer y escribir en los bloques.

2.6 Modos de ejecución

La mayoría de los procesadores dan soporte para dos modos de ejecución. Y estos modos son: el modo privilegiado que también se le conoce como modo del núcleo ó kernel; y él modo menos privilegiado que se le conoce como modo de usuario.

En el modo del núcleo, solo ciertas instrucciones pueden ser ejecutadas; como la lectura y modificación de registros de control (como la palabra de estado del programa), instrucciones de E/S e instrucciones relativas a la gestión de memoria.

La razón por la que existen dos modos de ejecución es para proteger al sistema operativo de los datos del usuario. En el modo del núcleo, se tiene control completo del procesador, de sus instrucciones, registros y memoria.

Hay un bit en la PSW (Program Status Word) que indica el modo de ejecución. Este es cambiado como respuesta a ciertos sucesos; por ejemplo cuando un usuario hace una petición a un servicio del sistema operativo, este cambia al modo del núcleo. Esto se hace ejecutando una instrucción que cambia el modo (CHM, *Change Mode*). Cuando el usuario hace una llamada a un servicio del sistema o cuando una interrupción transfiere el control a una rutina del sistema, la rutina ejecuta CHM para entrar en un modo más privilegiado y la ejecuta de nuevo para pasar a un modo menos privilegiado, antes de devolver el control al proceso del usuario. Si un programa de usuario intentará

ejecutar CHM, se originará una llamada al sistema operativo, que devolverá un error

Cuando un programa en modo usuario llama a un servicio del sistema, el procesador recoge la llamada y cambia a modo kernel (este cambio de modo de ejecución se conoce como *llamadas al sistema*, y son realizadas mediante una función *trap* o mediante alguna otra instrucción de lenguaje máquina). Cuando el servicio acaba, el sistema operativo vuelve a cambiar el contexto de la llamada a modo usuario y permite seguir funcionando al programa.

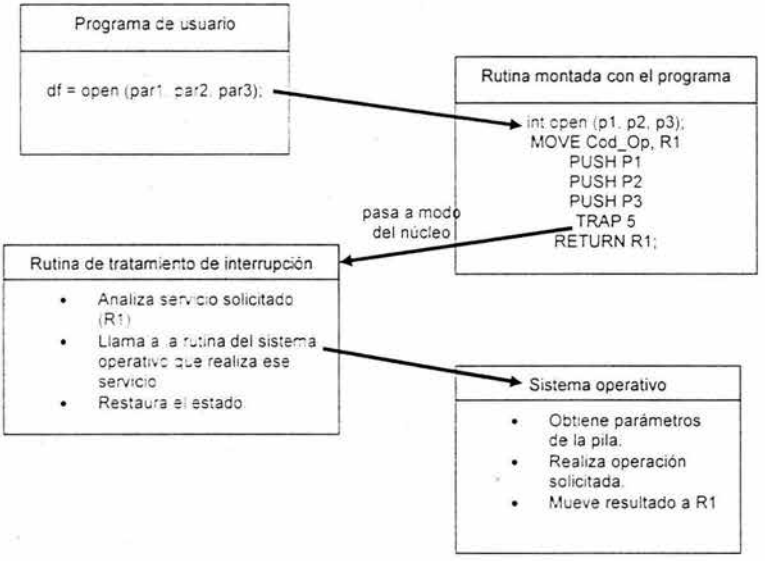


Figura 2.6 La figura muestra un ejemplo de cómo se realiza una llamada al sistema.

El sistema operativo, ofrece bibliotecas (rutinas) para poder realizar llamadas al sistema desde un lenguaje de programación sin tener que utilizar lenguaje ensamblador ni conocer el mecanismo de llamadas al sistema que utiliza el sistema operativo; la figura 2.6 muestra un ejemplo a bloques de lo que acabamos de describir.

2.7 Creación de procesos

Cuando se añade un proceso más a los que ya está administrando el sistema operativo, hay que construir las estructuras de datos que se utilizan para administrar el proceso (como se describe en la sección 2.3) y asignar el espacio de direcciones que va a utilizar el proceso. Estas acciones forman la creación de un proceso.

Existen cuatro sucesos comunes que llevan a la creación de procesos y estos se muestran a continuación:

Tabla 2.3 Razones para la creación de procesos.[12]

Suceso	Descripción
Nuevo trabajo por lotes	Un proceso se crea como respuesta a la remisión de un trabajo.
Conexión interactiva	Se crea un proceso cuando un usuario se intenta conectar desde una terminal.
Creado por el sistema operativo para dar un	Si un usuario solicita la impresión de un archivo, el sistema operativo creará un proceso

[12] Idem

servicio	que gestionará dicha impresión.
Generado por un proceso existente	Con afán de modularidad o para aprovechar el paralelismo, un programa de usuario puede ordenar la creación de una serie de procesos.

Una vez creado un proceso, el sistema operativo procede como sigue:

1. *Le asigna un identificador numérico único al proceso.* En ese momento se agrega una entrada nueva a la tabla de procesos, que contiene una entrada por proceso.

2. *Se asigna espacio para el proceso.* Esto incluye a todos los elementos de la imagen del proceso. El sistema operativo necesita saber cuánto espacio se necesitará para el espacio de direcciones del usuario y para la pila del usuario. Además, tiene que asignar espacio para el bloque de control del proceso.

3. *Inicializar el bloque de control del proceso.* La identificación del proceso contiene el ID de este proceso junto a otros ID's, tales como el del proceso padre. La parte de información del estado del procesador normalmente se inicializa con la mayor parte de las entradas a cero, excepto para el contador de programa (que se prepara con el punto de entrada del programa) y los apuntadores a las pilas del sistema (que establecen los límites de la pila del proceso). La parte de información de control del

procesador se inicializa a partir de los valores estándares por omisión y los atributos que se han solicitado para el proceso.

4. *Establecer los enlaces apropiados.* El sistema operativo mantiene cada cola de planificación como lista enlazada, entonces cada proceso se debe poner en la cola de Listos.

5. *Generar estructuras de datos para crear o ampliar.* Mantener un archivo de contabilidad de cada proceso que sea utilizado con propósito de evaluación de rendimiento.

2.8 Terminación de procesos

Así como existen varias formas de crear procesos, también existen formas de terminarlos. En un trabajo por lotes debe existir una instrucción *halt* que genere una interrupción para avisar al sistema operativo la terminación de un proceso. En una aplicación interactiva el usuario es quien termina el proceso. Estos son dos ejemplos que pueden terminar un proceso, sin embargo existen más (tabla 2.4), tales como una serie de condiciones o fallos que pueden llevar a la terminación de un proceso.

Tabla 2.4 Causas para la terminación de un proceso.[13]

Causa	Descripción
Terminación normal	El proceso ejecuta una llamada a un servicio del

[13] Idem

	sistema operativo que indica que ha terminado.
Tiempo límite excedido	El proceso ha estado en ejecución más tiempo que el total especificado.
No hay memoria disponible	El proceso necesita más memoria de la que el sistema operativo le puede proporcionar.
Violación de límites	El proceso intenta acceder a una posición de memoria que no tiene permitido acceder.
Error de protección	El proceso intenta utilizar un recurso que no tiene permitido, o no lo esta utilizando de manera correcta.
Error aritmético	Se intenta hacer un cálculo prohibido.
Tiempo máximo de espera rebasado	Ha esperado más allá del tiempo máximo establecido para que se produzca cierto suceso.
Fallo de E/S	Se produce un fallo de E/S, por la incapacidad de encontrar un archivo, o por un fallo de lectura o escritura.
Instrucción inválida	Cuando se intenta ejecutar una instrucción inexistente.
Instrucción privilegiada	Cuando se intenta usar una instrucción reservada para el sistema operativo.
Mal uso de los datos	Cuando un tipo de dato no es el adecuado o no está inicializado.
Intervención de operador o del sistema operativo	Por alguna razón el sistema operativo termina con el proceso.
Terminación del	Todos los procesos hijos deben terminar

padre	cuando termine el proceso padre.
Solicitud del padre	Un proceso padre tiene la autoridad para terminar con cualquiera de sus procesos hijos.

2.9 Cambio de proceso

A simple vista cambiar de proceso parece fácil; un proceso que se está ejecutando se interrumpe y el sistema operativo pone a otro en el estado de ejecución y pasa el control a ese proceso. Pero no es así de simple, veamos que sucede.

2.9.1 Cuando cambiar de proceso

En cualquier momento en que el sistema operativo tenga el control a partir del proceso que está ejecutándose, se puede dar un cambio de proceso. Para hacer esto, hay que tener en cuenta las interrupciones del sistema; de las cuales se encuentran principalmente dos tipos: una conocida como interrupción y otra como cepo.

Tabla 2.5 Mecanismos para la interrupción de la ejecución de un proceso.[14]

Mecanismo	Causa	Uso
Interrupción	Externa a la ejecución de la instrucción en	Reacción a un suceso asíncrono externo.

[14] Idem

	curso.	
Cepo	Asociada con la ejecución de la instrucción en curso.	Tratamiento de un error o de una condición excepcional
Llamada al supervisor	Solicitud explicita.	Llamada a una función del sistema operativo.

La interrupción es producida por algún evento externo al proceso. En éstas, el control se transfiere a un gestor de interrupciones, el cual realiza algunas funciones básicas, y después, brinca a una rutina del sistema operativo la cual se encarga del tipo de interrupción que se originó. Por ejemplo:

- *Interrupción de reloj:* el sistema determina si el proceso que ha estado en ejecución se encuentra dentro del tiempo máximo permitido. Esto provoca que el proceso pase al estado de Listo y se debe expedir otro proceso.

- *Interrupción de E/S:* el sistema detecta que se ha producido una acción de E/S. Si la acción constituye un suceso que están esperando uno ó más procesos, entonces, todos los procesos que estaban bloqueados pasan al estado Listo. El sistema decide si reanudar la ejecución del proceso que está actualmente en Ejecución o se expulsa a dicho proceso a favor de un proceso Listo.

- *Fallo de memoria:* cuado se hace referencia a una dirección de memoria virtual que no está en memoria principal. Esto es, cuando

se tiene que traer un bloque de memoria (página o segmento) que hace referencia de la memoria secundaria a la principal. Después que se genera una interrupción de E/S para traer un bloque de memoria, el sistema operativo lleva a cabo un cambio de contexto para reanudar la ejecución de otro proceso; el proceso que cometió el fallo se pasa al estado Bloqueado. Después de que el bloque se cargue en memoria, dicho proceso se pondrá en estado de Listo.

El cepo es producido por una condición de error o excepción dentro del proceso. En estos, el sistema operativo determina si ocurrió un error fatal. Si sucedió, el proceso pasa al estado Terminado y cambia de proceso. Si no ocurrió, la acción a seguir depende del tipo de error que se haya producido; se intentará una recuperación; como un cambio de proceso o reanudar el proceso, o tal vez sólo se le avise al usuario.

Una llamada del supervisor se produce desde el programa que está ejecutándose. Cuando un proceso de usuario esta ejecutándose y llega una instrucción que solicita una operación de E/S, como abrir un archivo; esta llamada provoca una transferencia a una rutina que forma parte del código del sistema operativo. El uso de una llamada al sistema hace que el proceso de usuario pase al estado de Bloqueado.

2.9.2 Cambio de contexto

Antes de abordar el tema es necesario hacer mención de cómo trabajan las interrupciones. Con estas, el procesador se puede dedicar

a la ejecución de otras instrucciones mientras una operación está en proceso. Cuando un dispositivo esté disponible, es decir, cuando esté preparado para aceptar más datos del procesador, el dispositivo enviará una señal de *solicitud de interrupción* al procesador. El procesador responde suspendiendo la operación del programa en curso y saltando a un programa que da servicio al dispositivo en particular, conocido como *rutina de tratamiento de la interrupción (interrupt handler)*, reanudando la ejecución en donde se había quedado después de haber atendido al dispositivo. Desde el punto de vista del usuario, una interrupción es solamente eso: una interrupción de la secuencia normal de ejecución. Cuando el tratamiento de la interrupción termina, la ejecución continúa.

Para dar cabida a las interrupciones, se añade un *ciclo de interrupción* al ciclo de instrucción. En el ciclo de interrupción, el procesador comprueba si ha ocurrido alguna interrupción, lo que indicará con la presencia de una señal de interrupción. Si no hay interrupciones pendientes, el procesador sigue con el ciclo de lectura y trae la próxima instrucción del programa en curso. Si hay una interrupción pendiente, el procesador suspende la ejecución del programa en curso y, volviendo al tema principal, hace lo siguiente:

1. Salva el contexto del programa que está ejecutándose.

2. Asigna al contador de programa, el valor de la dirección de comienzo de un programa de *tratamiento de la interrupción*. La rutina de tratamiento de la interrupción forma parte del sistema

operativo. Este programa determina la naturaleza de la interrupción y realiza las acciones necesarias.

El procesador continúa entonces con el ciclo de lectura de instrucción y trae la primera instrucción del programa de tratamiento de interrupciones, que atenderá a la interrupción.

El contexto del programa, incluirá toda aquella información que pueda ser alterada por la ejecución de la rutina de tratamiento de la interrupción y que pueda ser necesaria para reanudar el programa que fue interrumpido. Esto quiere decir que, debe guardarse la parte del bloque de control del proceso que se ha denominado información de estado del procesador; esto incluye el contador de programa (PC), otros registros del procesador y la información de la pila (*stack*).

Es posible que, después de que el gestor de interrupciones haya ejecutado el proceso que estaba ejecutándose, reanude su ejecución. Si sucede esto, lo único que hay que hacer es salvar la información de estado del procesador cuando se produzca la interrupción y restaurar dicha información cuando el control vuelva al programa que estaba en ejecución. Las funciones de *salvar y restaurar* se llevan a cabo en el hardware.

2.9.3 Cambio de estado

Es importante, no confundir un cambio de contexto con un cambio de proceso, ya que puede producirse un cambio de contexto sin

cambiar el estado del proceso que esta en ejecución. Si el proceso que se está ejecutando tiene que pasar a otro estado, el sistema operativo debe realizar cambios significativos en su entorno. Los pasos a seguir son los siguientes:

1. Salvar el contexto del procesador, incluyendo el contador del programa y otros registros.

2. Actualizar el bloque de control del proceso que estaba en Ejecución. Esto tiene como consecuencia cambiar el estado del proceso a alguno de los otros estados (Listo, Bloqueado, Terminado).

3. Mover el bloque de control del proceso a la cola apropiada (Listos, Bloqueados).

4. Seleccionar otro proceso para ejecución.

5. Actualizar el bloque de control de proceso seleccionado. Esto incluye cambiar el estado del proceso a Ejecución.

6. Actualizar las estructuras de datos de gestión de memoria.

7. Restaurar el contexto del procesador que existía en el momento en que el proceso seleccionado dejó por última vez el estado de Ejecución, cargando los valores previos del contador de programa y otros registros.

Como podemos ver, el cambio de proceso, que supone un cambio de estado, requiere un esfuerzo mucho mayor que un cambio de contexto.

2.10. Planificación del procesador

La asignación del procesador a los procesos hace posible que éstos realicen su trabajo, y tal asignación es un trabajo complejo que realiza el sistema operativo, esto se conoce como planificación del procesador, de la cual solo vamos a mencionar algunos puntos.

2.10.1 Planificador y activador

El afán de la planificación del procesador consiste en asignar los procesos al procesador para que sean ejecutados en algún momento, de forma que se cumplan objetivos del sistema tales como tiempo de respuesta, la productividad y la eficiencia del procesador. La clave de la multiprogramación está en la planificación.

El planificador forma parte del núcleo del sistema operativo. Entra en ejecución cada vez que se activa el sistema operativo, y básicamente su misión es seleccionar el proceso que se ha de ejecutar a continuación.

El activador también forma parte del sistema operativo y su función es poner en ejecución el proceso seleccionado por el planificador. La activación del sistema operativo se realiza mediante el

mecanismo de interrupciones. Cuando se produce una interrupción, se realizan las operaciones siguientes:

- Se salva el estado del procesador en el correspondiente BCP (bloque de control de proceso).

- Se pasa a ejecutar la rutina de tratamiento de interrupciones del sistema operativo.

En otras palabras, se hace un cambio de contexto.

2.10.2 El cronómetro de intervalos o reloj de interrupciones

Se dice que el proceso que está asignado al procesador, está en ejecución. Si el proceso pertenece al sistema operativo, se dice que el sistema operativo está en ejecución y puede tomar decisiones que afectan la operación del sistema. Para evitar que el usuario monopolice el sistema (ya sea deliberada o accidentalmente), el sistema operativo tiene mecanismos para arrebatar el procesador al usuario.

El sistema operativo mantiene un reloj de interrupciones o cronómetro de intervalos para generar interrupciones en algún momento futuro específico (o después de cierto tiempo). Luego, el procesador despacha al proceso. Éste, mantiene el control del procesador hasta que lo libera voluntariamente, hasta que el reloj interrumpe o hasta que alguna otra interrupción desvía la atención del

procesador. Si el usuario se encuentra en ejecución y el reloj interrumpe, el sistema operativo entra en ejecución y entonces decide a qué proceso se asignará en seguida el procesador.

El reloj de interrupciones ayuda a garantizar tiempos de respuesta aceptables para los usuarios interactivos, evita que el sistema quede bloqueado en un ciclo infinito de algún usuario y permite que los procesos respondan a *eventos dependientes del tiempo*. Los procesos que deben ejecutarse periódicamente dependen del reloj de interrupciones.

CAPÍTULO 3. EL HARDWARE

El punto que más nos interesa aquí, es describir de forma detallada como es cargado el sistema operativo. Para lo cual vamos a describir algunos puntos de hardware que se consideran importantes y que tienen relación directa con el proceso que sigue una computadora hasta que es montado el sistema operativo.

De manera muy básica podemos decir que, el hardware de una computadora se divide en tres partes: el cpu, la memoria y los dispositivos de E/S; esto se muestra en el diagrama a bloques siguiente:

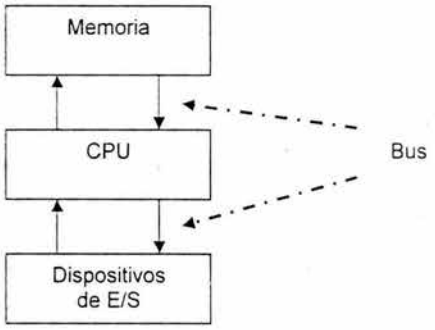


Figura 3.0 Diagrama general a bloques de una computadora.

El cpu es un circuito que interpreta y ejecuta instrucciones. Se ocupa del control y el proceso de los datos en las computadoras. Generalmente, el cpu es un microprocesador fabricado en un chip de

silicio que contiene millones de componentes electrónicos. Está formado por una unidad aritmético y lógica que realiza cálculos, comparaciones y toma decisiones lógicas, además contiene una serie de registros donde se almacena información temporalmente y, también esta constituido, por una unidad de control que interpreta y ejecuta las instrucciones. Para aceptar órdenes del usuario, acceder a los datos y presentar los resultados, se comunica a través de un conjunto de circuitos o conexiones llamado bus.

La memoria almacena los datos e instrucciones de uso más frecuente. Hay que decir que existen varios tipos de memoria, como por ejemplo la RAM (Random Access Memory) o la Cache. La memoria está organizada en celdas, como una hoja cuadriculada, y para acceder a una celda determinada se utiliza el número de fila y el número de columna.

Los dispositivos de E/S incluyen teclados, impresoras, dispositivos de comunicación entre otros. Estos dispositivos, facilitan la comunicación de la computadora con el usuario, permitiendo el ingreso y la salida de información de la computadora.

Si vemos a la computadora desde el punto de vista del usuario final podemos decir que, es una máquina que puede resolver problemas ejecutando las instrucciones que recibe del usuario.

Pero esto no es tan simple como parece; los circuitos de la computadora pueden reconocer y ejecutar, de manera transparente,

solo un conjunto de las instrucciones de usuario: es por esto que todos los programas tienen que ser convertidos, pasando por intérpretes, traductores, compiladores, hasta llegar a un conjunto de instrucciones especiales para que la computadora pueda ejecutarlos. Estas instrucciones especiales a las que nos estamos refiriendo por lo regular no hacen más que[15]:

- Sumar dos números
- Copiar un dato
- Verificar si un número es cero

A este conjunto de instrucciones, que permiten que nos comuniquemos con la computadora, las conocemos con el nombre de lenguaje máquina.

3.1 La computadora y los niveles

La primera generación de computadoras (aproximadamente 1937-1949) contaba con tan solo dos niveles; el de lógica digital (hardware) y el ISA (arquitectura del conjunto de instrucciones). Los programas escritos en esta época eran escritos en lenguaje máquina y se ejecutaban directamente por el hardware, sin la intervención de compiladores, traductores o interpretes. En 1949

[15] Tanenbaum S. Andrew. **Organización de computadoras. Un enfoque estructurado.** Cuarta Edición. Prentice Hall. México. 2000. 688 páginas.

aparece la microprogramación, termino atribuido al profesor Maurice Wilkes, quien utilizó un nivel más en el desarrollo de EDSAC. Alrededor de los años 60's aparece otro nivel, el cual estaba ocupado por un programa que esta cargado todo el tiempo en memoria, el sistema operativo. En la actualidad la mayoría de las computadoras cuenta con una serie de niveles y pueden llegar a existir hasta seis niveles. Tal y como se indican en la figura 3.1.

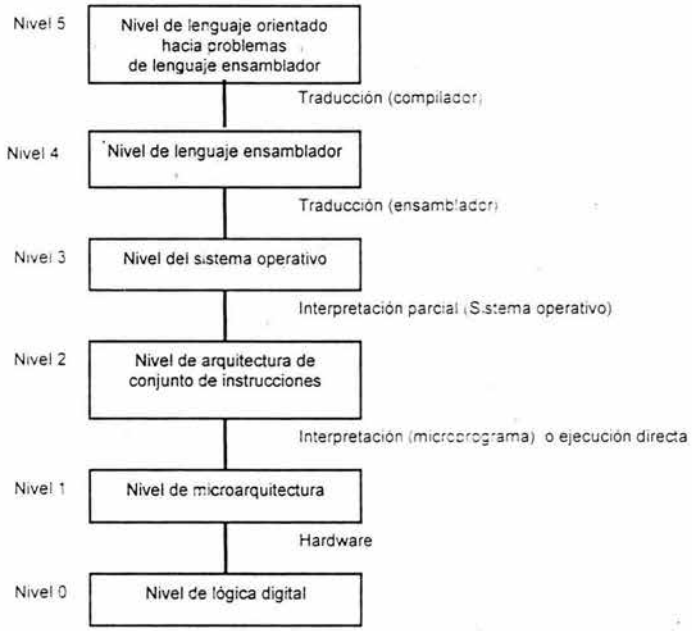


Figura 3.1 Computadora de seis niveles. El método de apoyo para cada nivel se indica inmediatamente debajo de él (junto con el nombre del programa de apoyo)[16]

[16] Idem.

A continuación describiremos cada uno de estos niveles.

3.1.1 Nivel de lógica digital

En este nivel se encuentra el hardware. La primer computadora que contaba con este nivel fue ENIAC (Electronic Numerical Integrator and Computer), construida por John W. Mauchley y John P. Eckert en 1943. Esta computadora era programada por un conjunto de interruptores y cables que se conectaban y desconectaban según lo que se quisiera hacer.

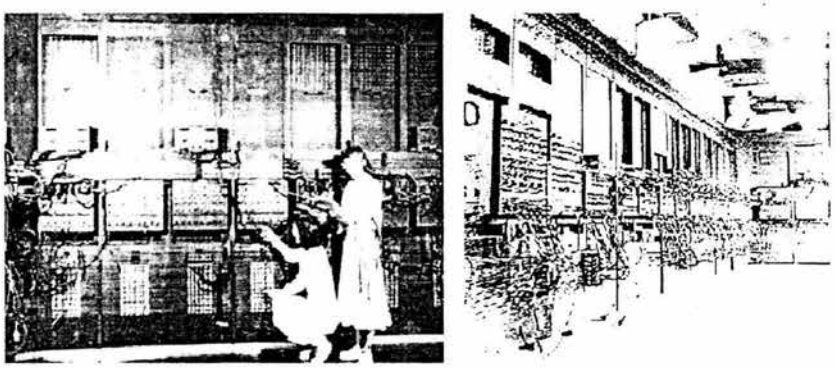


Figura 3.1.1 Ejecutando un Programa en ENIAC

En los años 50's, aparecen las primeras computadoras digitales de propósito general. Éstas usaban tubos al vacío (bulbos) como componentes electrónicos activos. Tarjetas o módulos de tubos al vacío fueron usados para construir circuitos lógicos básicos tales como compuertas lógicas y flip-flops. Ensamblando compuertas y flip-flops en

módulos, los científicos construyeron, de manera básica, la computadora (la lógica de control, memorias, entre otros). Los bulbos también formaron parte de la construcción de máquinas para la comunicación con las computadoras.

La construcción de una computadora digital requiere de muchos circuitos o dispositivos electrónicos. El principal paso fue hacer que el dato fuera almacenado en memoria, como una forma de palabra digital. La idea de almacenar programas fue muy importante.

La tecnología de los circuitos de estado sólido evolucionó en la década de los años 50's. El uso de un material llamado silicio de bajo costo combinado con métodos de producción masiva, dieron paso a la aparición del transistor (creado en 1948 en los laboratorios Bell), el cual paso a ser el elemento más usado para el diseño de circuitos. Por lo tanto, el diseño de la computadora digital fue un gran avance en el proceso del cambio para remplazar al tubo al vacío (bulbo) por el transistor a finales de los años 50's.

A principios de los años 60's, el arte de la construcción de computadoras de estado sólido se incrementó y surgieron las tecnologías en circuitos digitales como: RTL (Lógica Transistor Resistor), DTL (Lógica Transistor Diodo), TTL (Lógica Transistor Transistor), ECL (Lógica Complementada Emisor).

A mediados de los años 60's se producen las familias de lógica digital, dispositivos en escala SSI (Small Scale Integrated) que maneja

de 1 a 10 compuertas, MSI (Médium Scale Integrated) que maneja de 10 a 100 compuertas. A finales de los años 60's y principios de los años 70's surgieron los LSI (Large Scale Integrated) la cual maneja de 100 a 100000 compuertas. La tecnología LSI hizo posible la integración en un solo chip de circuitos digitales. Pero pocos circuitos LSI fueron producidos, los dispositivos de memoria fueron un buen ejemplo. Después aparecen los VLSI (Very Large Scale Integrated) que utilizan más de 100000 compuertas.

3.1.2 Nivel de microarquitectura.

Este nivel es el único que existía por encima del hardware. Aquí todo se daba a nivel de transferencia de registros y no a nivel de compuertas como en el nivel de hardware.

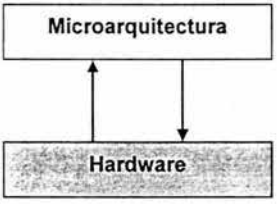


Figura 3.1.2 El único nivel por encima del hardware era el de microarquitectura.[17]

Los microprogramas están formados por microinstrucciones que son mucho más elementales, en naturaleza y alcance, que las

[17] Idem.

instrucciones en lenguaje máquina; un microprograma es software que generalmente se localiza en una memoria de solo lectura, y su función es ejecutar las instrucciones del lenguaje máquina en forma secuencial. Dicho de otra manera, en una computadora microprogramada, las instrucciones del nivel de máquina convencional, como llamadas a procedimientos, multiplicaciones e iteraciones, no son efectuadas directamente por el hardware, sino que son extraída, examinadas y ejecutadas por el microprograma como una serie de pequeños pasos.

Las microinstrucciones se pueden clasificar en horizontales y verticales. Las microinstrucciones verticales proporcionan un control explícito de las funciones en puntos determinados dentro de la unidad central de proceso. Por ejemplo; un bit determinado en una microinstrucción, podría exigir la puesta a cero de un registro determinado en un tiempo específico de reloj.

Generalmente, las microinstrucciones verticales contienen campos codificados y describen operaciones a ser realizadas por ciertos elementos de la unidad de control, la unidad aritmética y lógica, y por las fuentes y destinos de información que pasan entre estas unidades. Su ejecución es muy parecida con las instrucciones en lenguaje máquina. Una microinstrucción vertical típica especifica el movimiento de un campo entre registros.

El microcódigo horizontal es bastante diferente. Cada instrucción necesita más bits para especificar la operación del movimiento paralelo de datos entre los registros de datos de la unidad de control. Estas

microinstrucciones son más poderosas que las verticales, pero los programas resultantes son también más dificiles de codificar y depurar.

3.1.2.1 Microprogramación

En mayo de 1949 aparece EDSAC (Electronic Delay Storage Automatic Computer) y con ella la microprogramación. El diseño de está máquina se acredita a Maurice Wilkes y sus colegas en el laboratorio de Matemáticas de la Universidad de Cambridge, la cual se utilizó principalmente como una herramienta para matemáticos. La máquina contaba con pantallas de tubo de rayos catódicos para mostrar las operaciones electrónicas del hardware con propósitos de diagnóstico, así como tanques para las líneas de retraso de mercurio que eran utilizadas para el almacenamiento principal.

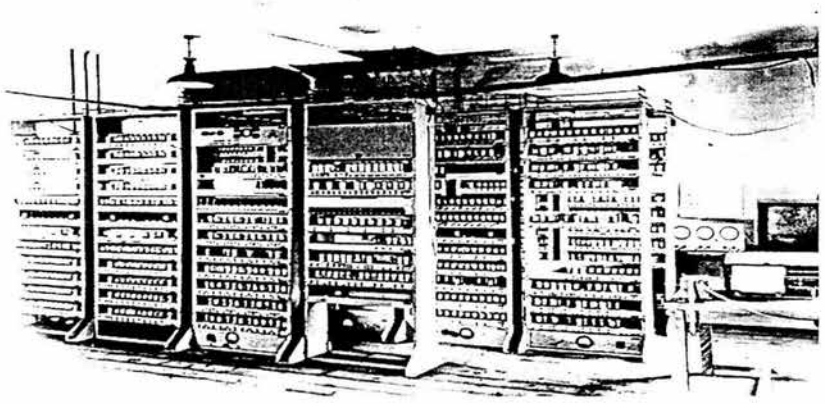


Figura 3.1.2.1 La figura nos muestra una idea de lo que era EDSAC

Su principio de funcionamiento era el de convertir pulsos eléctricos a pulsos ultrasónicos que se propagaban a través de un tubo de cristal lleno de mercurio desde un extremo al otro. Cuando llegaban al extremo se reconvertían a señal eléctrica. En el EDSAC se disponía de 16 tanques de mercurio capaces de almacenar 256 palabras de 35 bits o 512 palabras de 16 bits. La velocidad de reloj del EDSAC era de 500 khz y la mayoría de las instrucciones tardaban en ejecutarse 1.5 segundos, una eternidad en comparación con las computadoras actuales. Las entradas y salidas de datos se hacen mediante cinta de papel.

Esta máquina contaba con un nivel más (tres en total en ese momento); Wilkes definió una librería de pequeños programas llamados subrutinas, almacenadas en cintas de papel perforado las cuales, con el tiempo, se conocieron como microprograma (un interprete), el cual solamente tendría que ejecutar las instrucciones del nivel ISA. Esto quiere decir que el hardware tendría que ejecutar solamente microprogramas que son un conjunto limitado de instrucciones, en lugar de ejecutar programas del nivel ISA que tienen un conjunto de instrucciones más grande, esto significo un ahorro en hardware considerable.

La microprogramación tuvo su mayor desarrollo a partir de los años 60's. A principios de los 70, apareció la microprogramación dinámica que permite a partir de microprogramas en ejecución la carga de nuevos microprogramas en la memoria, con lo que el conjunto de instrucciones máquina puede variarse dinámica y frecuentemente.

En la decada de los 80´s empieza su decadencia con la aparición de la arquitectura RISC (Reduced Instruction Set Computing). Esto se debe a que, como su nombre lo indica, esta arquitectura trabaja con un conjunto muy reducido de instrucciones en comparación con la arquitectura CISC; además de que el cpu ejecuta las instrucciones en menos ciclos de reloj, esto se debe a que, originalmente, las instrucciones de la arquitectura RISC solamente necesitaban un ciclo de reloj por instrucción, lo que tiene como consecuencia una mayor velocidad y eficiencia, entre otras características.

En este nivel es difícil distinguir que es electrónica y que es programación ya que la frontera entre el hardware y el software es muy delgada. Es por ello que a los microprogramas también se les conoce como firmware.

3.1.2.2 La microprogramación y los sistemas operativos

En la mayoría de los sistemas operativos, ciertos componentes están en la secuencia de instrucciones que se ejecutan con más frecuencia. Por ejemplo; en un sistema de procesos el mecanismo de "dispatching" (despachador de procesos) con el que se selecciona el siguiente proceso que ha de hacer uso del microprocesador, podría ser ejecutado a cientos de veces por segundo. Esto obliga a que este mecanismo funcione eficientemente; una forma de acelerarlo es ponerlo en microcódigo. Así, algunas de las funciones que suelen implementarse con microcódigos son:

- Gestión de interrupciones.

- Mantenimiento de diferentes tipos de estructuras de datos (pilas, listas encadenadas).

- Primitivas de sincronización que controlan el acceso a datos y recursos compartidos.

- Operaciones de palabra parcial que permiten la manipulación de bits eficientemente.

- Conmutación de contexto; es decir, la rápida conmutación del procesador entre los usuarios de un sistema multiusuario.

- Secuencias de llamada y retorno de procedimiento.

En consecuencia, desarrollando funciones del sistema operativo en microcódigo se consigue mejorar el rendimiento, reducir los costos de desarrollo de programas y mejorar la seguridad del sistema.

3.1.3 Nivel de arquitectura del conjunto de instrucciones (ISA)

Este nivel fue el primero que se desarrollo, a decir verdad era el único que existía por encima del hardware. Tiene sus orígenes con el asesoramiento que John Louis Von Neumann (1903-1957) prestó a Eckert y Mauchly en el diseño y fabricación de ENIAC (1944), en donde

aplicó por vez primera lo que hoy conocemos como arquitectura de Von Neumann.

El ISA es un nivel de alta importancia por la sencilla razón de que es la interfaz entre el software y el hardware. Para producir código en el nivel ISA, el desarrollador tiene que conocer el modelo de memoria, los registros disponibles, los tipos de datos e instrucciones con que se cuenta, entre otras cosas. El conjunto de toda esta información es lo que define el nivel ISA; que no son más que un conjunto de instrucciones que son ejecutadas por un microprograma o por los circuitos de ejecución de hardware.

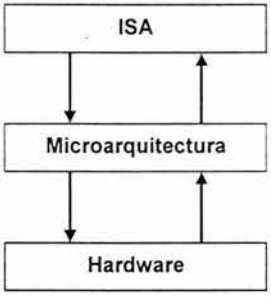


Figura 3.1.3 Aparece el nivel de arquitectura del conjunto de instrucciones (ISA)[18]

El ISA nos proporciona un conjunto de instrucciones que se pueden utilizar con gran eficiencia en tecnologías actuales y futuras. Esto quiere decir que un usuario cualquiera puede ejecutar software

[18] Idem.

reciente (con ciertas limitaciones) en la computadora que compro hace algunos años. Esto permite que se pueda sacar el mayor provecho al hardware y al software, tanto en el aspecto económico como en capacidad de cómputo.

En resumen podemos decir que, el nivel ISA es la interfaz entre el hardware y el software, entonces, debe poder complacer a los diseñadores de hardware, a los de software y más que a nadie, al usuario final.

3.1.4 Nivel del sistema operativo

Aparece una cantidad importante de instrucciones, además de las instrucciones del nivel ISA, que pueden usar los programadores; este conjunto de instrucciones son conocidas como *llamadas al sistema*.

Entre el conjunto de cambios que resaltan podemos decir que hay una organización distinta de la memoria, aparece el procesamiento en paralelo (la multiprogramación), la memoria virtual que proporciona un espacio de direcciones mayor que la memoria física de una computadora y el manejo de la E/S de los archivos es más eficiente.

Como ejemplos de este conjunto de instrucciones podemos decir que, los que manejan UNIX le llaman POSIX (Portable Operating System IX), que es un estándar internacional conocido como IEEE

P1003, y los que manejan Windows recibe el nombre de Win32 proporcionado por Microsoft pero hay que decir que no es un estándar.

Las nuevas instrucciones son procesadas por un intérprete, el cual es conocido como sistema operativo.

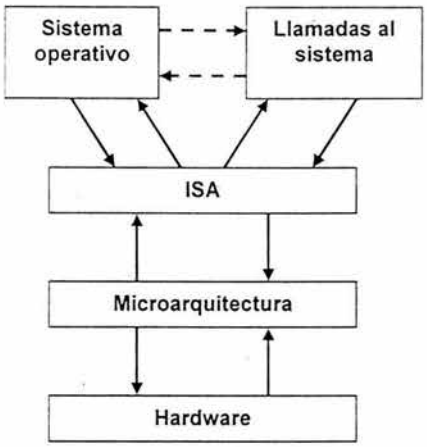


Figura 4.1.4 El nivel del sistema operativo aparece cuando se agregan nuevas instrucciones al nivel ISA, esta instrucciones se conocen como *llamadas al sistema.*[19]

El microprograma ejecuta directamente las instrucciones del nivel tres que son idénticas a las del nivel dos. Dicho de otra manera, algunas de las instrucciones del nivel tres son interpretadas por el sistema operativo y otras por el microprograma.

[19] Idem.

3.1.5 El nivel de lenguaje ensamblador

Permite escribir programas en forma de palabras y abreviaturas fácilmente comprensibles para el hombre y después traducirlos a lenguaje máquina, que es el lenguaje que se maneja en los niveles uno, dos y tres. A este programa traductor se le conoce con el nombre de ensamblador.

La comunicación en lenguaje de máquina es particular de cada procesador que se usa, y programar en este lenguaje es muy difícil y tedioso, por lo que se empezó a buscar mejores medios de comunicación con ésta.

A principios de la década de 1950, y con el fin de facilitar la labor de los programadores, se desarrollaron códigos mnemotécnicos para las operaciones y direcciones simbólicas. Uno de los primeros pasos para mejorar el proceso de preparación de programas fue sustituir los códigos de operación numéricos del lenguaje de máquina por símbolos alfabéticos, que conforman un lenguaje mnemotécnico. Todas las computadoras actuales tienen códigos mnemotécnicos aunque, naturalmente, los símbolos que se usan varían en las diferentes marcas y modelos. La computadora sigue utilizando el lenguaje de máquina para procesar los datos, pero los programas ensambladores traducen antes los símbolos de código de operación especificados a sus equivalentes en lenguaje de máquina.

Los lenguajes ensambladores tienen ventajas sobre los lenguajes de máquina. Ahorran tiempo y requieren menos atención a detalles. Se incurren en menos errores y los que se cometen son más fáciles de localizar. Además, los programas en lenguaje ensamblador son más fáciles de modificar que los programas en lenguaje de máquina. Pero existen limitaciones. La codificación en lenguaje ensamblador es todavía un proceso lento. Además, una desventaja importante de estos lenguajes es que tienen una orientación a la máquina. Es decir, están diseñados para la marca y modelo específico de procesador que se utiliza.

Figura 3.1.5 Con la aparición del lenguaje ensamblador se facilito la labor de programadores, la desventaja era que el leguaje está casado con la marca y modelo del microprocesador.

3.1.6 Nivel de lenguajes de alto nivel

El último nivel, consta de una gran cantidad de lenguajes, mejor conocidos como lenguajes de alto nivel. Los programas escritos en este tipo de lenguaje son traducidos, a los lenguajes de los niveles más bajos (nivel tres y cuatro), con traductores llamados compiladores, aunque ocasionalmente se interpretan en lugar de traducirse.

Los primeros programas ensambladores producían sólo una instrucción en lenguaje de máquina por cada instrucción del programa fuente. Para agilizar la codificación, se desarrollaron programas ensambladores que podían producir una cantidad variable de instrucciones en lenguaje de máquina por cada instrucción del programa fuente. Dicho de otra manera, una sola macroinstrucción podía producir varias líneas de código en lenguaje de máquina.

El desarrollo de las técnicas mnemotécnicas y las macroinstrucciones condujo, a su vez, al desarrollo de lenguajes de alto nivel que a menudo están orientados hacia una clase determinada de problemas de proceso.

A diferencia de los programas de ensamble, los programas en lenguaje de alto nivel se pueden utilizar con diferentes marcas de computadoras sin tener que hacer modificaciones considerables. Esto permite reducir sustancialmente el costo de la reprogramación cuando se adquiere equipo nuevo.

Otras ventajas de los lenguajes de alto nivel son:

- Son más fáciles de aprender que los lenguajes ensambladores
- Se pueden escribir más rápido
- Permiten mejor documentación

Figura 3.1.6 El desarrollo de los mnemonicos dan paso a la aparición de los lenguajes de alto nivel.

3.2 Los registros

De manera general, la función de los registros de una computadora es controlar la ejecución de un programa, almacenar resultados temporales, entre otras cosas. Los registros se dividen por lo general en dos categorías: registros de propósito especial y registros de propósito general.

Los registros de carácter especial incluyen al contador de programa y el apuntador a la pila, además de otros registros con funciones específicas. Los registros de propósito general sirven para guardar variables locales y los resultados intermedios de los cálculos. Su función es proporcionar acceso rápido a los datos que se usan continuamente.

Además de los registros que los programas de usuario pueden ver, siempre existen registros de propósito especial que sólo son accesibles en modo kernel. Estos registros controlan las diversas cachés, memoria, dispositivos de E/S y otras características del hardware de la máquina. Solamente el sistema operativo los usa, así que los compiladores y los usuarios no tienen que saber de ellos.

En este apartado vamos a hablar solamente de algunos registros de propósito especial que tienen relación directa con el sistema operativo, los registros que vamos a tratar son los soportados por la arquitectura Intel desde el 8086 hasta el Pentium II.

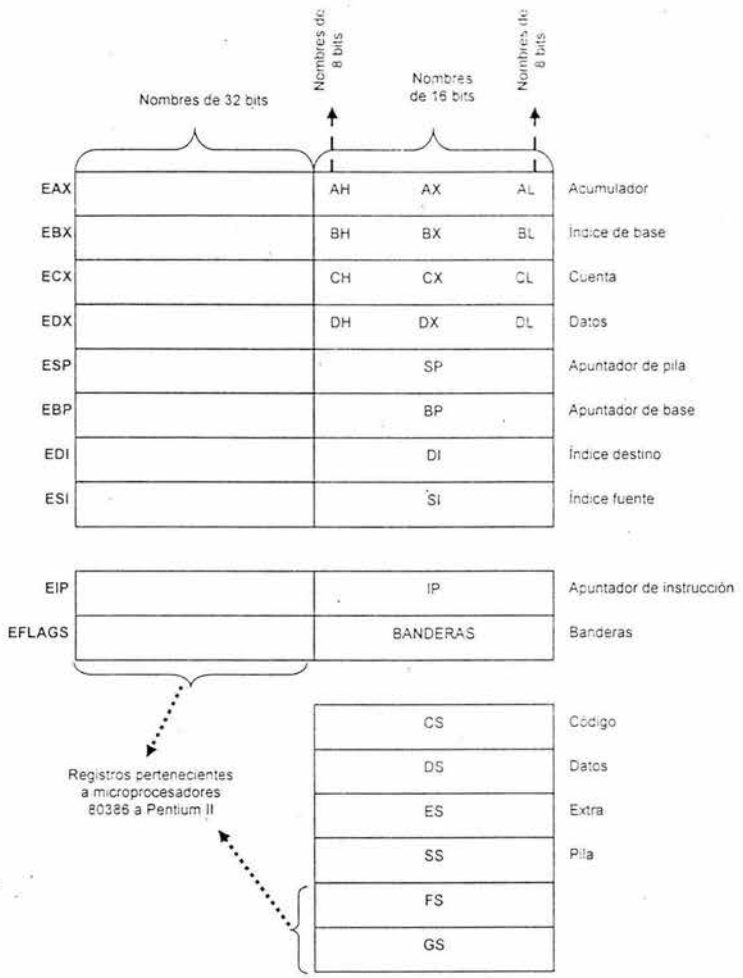


Figura 3.2 El modelo de programación de los microprocesadores Intel 8086 al Pentium II [20]

[20] Barry B. Brey. **Los microprocesadores Intel. Arquitectura y programación de los procesadores 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, Pentium Pro y Pentium II.** Quinta Edición. Pearson Educación. México, 2001. 976 Páginas.

En la figura 3.2 se muestra el modelo de programación de los microprocesadores del 8086 al Pentium II. Hay que resaltar que los modelos 8086, 8088 y 80286 poseen arquitecturas internas de 16 bits; los microprocesadores 80386, 80486, Pentium, Pentium Pro y Pentium II poseen arquitecturas internas completas de 32 bits. Además tenemos que decir, aunque es obvio, que las arquitecturas del 8086 al 80286 son compatibles con las arquitecturas del 80386 al Pentium II.

Las áreas sombreadas de la figura representan los registros que no se encuentran en los microprocesadores 8086, 8088 ó 80286, pero que representan una mejora en los microprocesadores 80386, 80486, Pentium, Pentium Pro y Pentium II.

El modelo de programación contiene registros de 8, 16 y 32 bits. Los registros de 8 bits son AH, AL, BH, BL, CH, CL, DH y DL y son especificados cuando una instrucción se forma utilizando estas denominaciones de dos letras.

Los registros de 16 bits también se representan con dos letras, como por ejemplo: AX, BX, CX, DX, SP, BP, DI, SI, IP, FLAGS, CS, DS, ES, SS, FS y GS.

Los registros extendidos de 32 bits son EAX, EBX, ECX, EDX, ESP, EBP, EDI, ESI, EIP y EFLAGS. Estos registros de 32 bits y los de 16 bits FS y GS, se encuentran disponibles solamente en los microprocesadores 80386 y posteriores.

A continuación vamos a describir algunos de los registros más importantes para el sistema operativo.

EIP (apuntador de instrucciones). El EIP direcciona la instrucción siguiente en una sección de memoria definida como segmento de código. Este registro es IP (16 bits) cuando el microprocesador opera en modo real, y EIP (32) bits cuando el 80386 o posterior operan en modo protegido. El microprocesador utiliza el apuntador de instrucciones para encontrar la siguiente instrucción secuencial en un programa ubicado dentro del segmento de código. El apuntador de instrucción puede ser modificado con una instrucción de salto ó de llamada de subrutina.

ESP (apuntador de pila). El ESP direcciona un área de memoria llamada pila. La memoria de pila almacena datos por medio de este apuntador; este registro es especificado como SP cuando se usa como registro de 16 bits y como ESP cuando se usa como registro de 32 bits.

Un registro importante a tratar es la palabra de estado del programa (PSW, Program Status Word) o registro de banderas. Este registro contiene diversos bits que el cpu necesita, pero los más importantes son los códigos de condición.

Estos bits se ajustan en cada ciclo de la ALU y reflejan la situación del resultado de la operación más reciente. La PSW contiene más que sólo los códigos de condición, pero el contenido total varía de máquina a máquina. Algunos otros campos importantes del PSW son el

modo máquina, un bit de rastreo (que sirve para depurar), el nivel de prioridad de la cpu y la habilitación de interrupciones.

EFLAGS (banderas). Indica la condición del microprocesador y controla su operación. La figura siguiente muestra los registros de banderas de todas las versiones del microprocesador. Hay que observar que los microprocesadores 8086 y 80286 contienen un registro FLAG (registro de banderas de 16 bits), mientras que el 80386 y posteriores contienen un registro EFLAGS (registro de 32 bits).

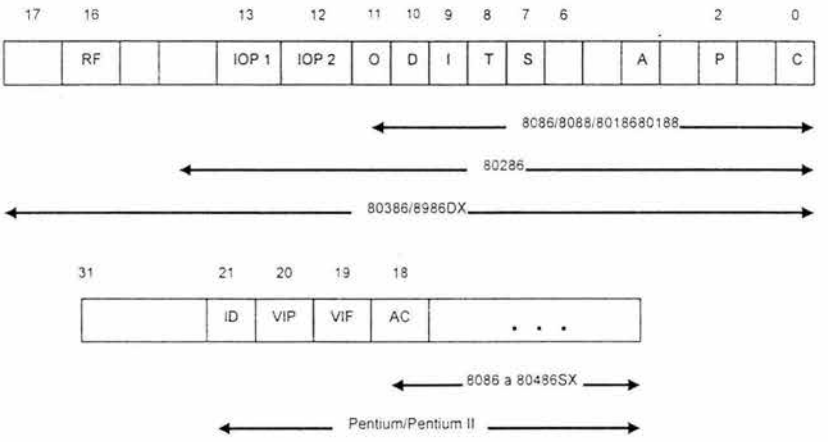


Figura 3.2.1 Las posiciones de las banderas en los registros FLAG y EFLAGS para la familia completa de microprocesadores 80x86 y Pentium.[21]

[21] Idem

Los cinco bits de bandera del extremo derecho y la bandera de desbordamiento cambian después de la ejecución de numerosas instrucciones lógicas y aritméticas. Las banderas nunca cambian por efecto de la transferencia de datos u operaciones de control del programa. Algunas de las banderas se usan para controlar características del microprocesador. A continuación se muestra una descripción de cada uno de los bits del registro de banderas.

C (acarreo). La bandera de acarreo contiene las unidades que se "llevan" después de una suma, o las que se "prestan" después de una resta. Esta bandera también indica condiciones de error, según lo establezcan algunos programas o procedimientos. Esto es especialmente válido para las llamadas de funciones del DOS.

P (paridad). La bandera de paridad toma el valor de cero lógico para paridad impar y de uno lógico para la par. La paridad es una cuenta de bits iguales a uno de un número, expresada como un valor par o impar. Hay pocas aplicaciones para la bandera de paridad en la programación moderna ya que fue incluida en los primeros microprocesadores de Intel con el fin de verificar datos en ambientes de comunicación de datos. Actualmente, son los equipos de comunicaciones los que a menudo llevan a cabo la verificación de la paridad, en lugar del microprocesador.

A (acarreo auxiliar). El acarreo auxiliar contiene el acarreo (medio acarreo) después de una suma o el préstamo después de una resta entre las posiciones de bit 3 y 4 del resultado. Esta bandera altamente

especializada es verificada por las instrucciones DAA y DAS para ajustar el valor del registro AL, después de una suma o una resta de cantidades expresadas en BCD. De no ser así, ni el microprocesador ni ninguna otra instrucción utilizan la bandera A para otro propósito.

Z (cero). La bandera cero indica que el resultado de una operación aritmética o lógica es cero. Si Z=1, el resultado es cero; si Z=0, el resultado es diferente de cero.

S (signo). Contiene el signo aritmético del resultado después de ejecutar una operación aritmética o lógica. Si S=1, el bit de signo (el bit del extremo izquierdo de un número) se encuentra activado o es negativo; si S=0, el bit de signo está inactivo o es positivo.

T (trampa). La bandera de trampa habilita la función de captura de errores por medio de una característica de depuración del dispositivo. (Un programa es depurado para encontrar un error o *bug*). Si la bandera T es habilitada (1), el microprocesador interrumpe el flujo de un programa bajo las condiciones especificadas en los registros de depuración y los de control. Si la bandera T está en cero lógico, la función de depuración se encuentra deshabilitada. El programa CodeView puede utilizar la función de depuración y los registros asociados para depurar programas defectuosos.

I (interrupción). Controla la operación de la terminal de entrada INTR (solicitud de interrupción). Si I=1, la terminal INTR está habilitada; si I=0, está deshabilitado. El estado del bit de bandera I es controlado

por las instrucciones STI (activar bandera de interrupción) y CLI (borrar bandera de interrupción).

D (dirección). La bandera de dirección selecciona ya sea el modo de incremento o el de decremento para los registro DI y SI durante las instrucciones en cadena. Si D=1, los registros disminuyen automáticamente; si D=0, los registros aumentan automáticamente. La bandera D se activa con la instrucción STD (activar bandera de dirección) y se desactiva con la instrucción CLD (borrar bandera de dirección).

O (desbordamiento). El desbordamiento puede ocurrir al sumar o restar números con signo. Un desbordamiento indica que el resultado ha excedido la capacidad de la máquina. Para operaciones sin signo la bandera de desbordamiento es ignorada.

IOPL (nivel de privilegio de E/S). En el modo de operación protegido, IOPL selecciona el nivel de privilegio para los dispositivos de E/S. Si el nivel de privilegio actual es mayor o más confiable que el IOPL, la E/S se ejecuta sin impedimento. Si el IOPL es menor que el nivel actual, ocurre una interrupción, ocasionando la suspensión de la ejecución.

NT (tarea anidada). La bandera de tarea anidada señala que la tarea actual está anidada dentro de otra, en el modo protegido de operación. Esta bandera se activa cuando la tarea es anidad por software.

RF (reanudar). La bandera de reanudación se utiliza durante la depuración para controlar la reanudación de la ejecución después de la siguiente instrucción.

VM (modo virtual). El bit de bandera VM selecciona la operación en modo virtual de un sistema que se encuentra en modo protegido. Un sistema en modo virtual permite la coexistencia en memoria de múltiples particiones del DOS con longitud de 1 MB cada una. En esencia, esto permite ejecutar múltiples programas del DOS.

AC (verificación de alineación). El bit de bandera de verificación de alineación se activa al direccionar una palabra o una doble palabra en una dirección que no es un número par (para palabras) o múltiplo de 4 (para palabras dobles). Sólo el microprocesador 80486SX contiene el bit de verificación de alineación utilizado principalmente por el coprocesador numérico y el 80487SX lo utiliza con propósitos de sincronización.

VIF (bandera de interrupción virtual). La bandera VIF es una copia del bit de bandera de interrupción que está disponible en los microprocesadores Pentium y Pentium II.

VIP (interrupción virtual pendiente). La bandera VIP proporciona información sobre una interrupción en modo virtual para los microprocesadores Pentium a Pentium II. Esta información es usada en ambientes multitarea para proporcionar al sistema operativo información

sobre banderas de interrupción virtuales e información sobre interrupciones pendientes.

ID (identificación). La bandera ID señala que los microprocesadores del Pentium al Pentium II aceptan la instrucción CPUID. La instrucción CPUID proporciona al sistema información sobre el microprocesador, tal como el número de versión y el fabricante.

Por otro lado, los registros de segmento son registros adicionales que generan direcciones de memoria al combinarse con otros registros del microprocesador. Un registro de segmento funciona de manera diferente dependiendo de si el microprocesador opera en modo real o en modo protegido. A continuación se muestran estos registros.

CS (código). El segmento de código es una sección de la memoria que contiene el código (programas y procedimientos) utilizado por el microprocesador. El registro del segmento de código define la dirección inicial de la sección de memoria que contiene el código. En operación de modo real, define el principio de una sección de 64 KB de memoria; en modo protegido, selecciona un descriptor que muestra la dirección de inicio y la longitud de una sección de memoria que contiene código. El segmento de código está limitado a 64 KB en los 8088 al 80286 y a 4 GB en los 80386 y posteriores, cuando estos microprocesadores operan en modo protegido.

DS (datos). Es una sección de memoria que contiene la mayoría de los datos utilizados por un programa. En este segmento, se accede a

los datos por medio de un desplazamiento o a través del contenido de otros registros que contienen la dirección de desplazamiento. Al igual que los segmentos de código y otros segmentos, su longitud está limitada a 64 KB en los 8086 al 80286 y a 4 GB en los 80386 y posteriores.

SS (pila). El segmento de pila define el área de memoria utilizada para la pila. El punto de entrada de la pila está determinado por los registros de este segmento, así como por los registros de los apuntadores de pila.

La memoria de pila juega un papel importante en todos los microprocesadores. Contiene temporalmente datos y almacena direcciones de retorno para procedimientos. La memoria de almacenamiento es de tipo LIFO (último en entrar, primero en salir) lo que describe la forma en son almacenados y extraídos los datos de la pila. Los datos son colocados en la memoria de pila con una instrucción PUSH y extraídos de ella con una instrucción POP. La instrucción CALL también utiliza la memoria de pila para contener la dirección de retorno para procedimientos y una instrucción RET para extraer la dirección de retorno de pila.

La memoria de pila es controlada por dos registros: el apuntador de pila (SS o ESP) y el registro de segmento de pila (SS).

3.3 Interrupciones

Las interrupciones son cambios en el flujo de control causados no por el programa en ejecución sino por otra cosa, casi siempre relacionada con dispositivos de E/S que proporcionan datos a velocidades de transferencia relativamente bajas. La interrupción detiene el programa en ejecución y transfiere el control a un manejador de interrupciones, el cual realiza alguna acción apropiada. Cuando el manejador de interrupciones termina, devuelve el control al programa interrumpido. El proceso interrumpido se debe reiniciar en el mismo estado exactamente en el que estaba cuando ocurrió la interrupción, lo que implica restaurar todos los registros internos a su estado previo a la interrupción.

Una petición de interrupción IRQ ("Interrupt Request") es una señal que se origina en un dispositivo hardware (por ejemplo, un periférico), para indicar al procesador que algo requiere su atención inmediata; se solicita al procesador que suspenda lo que está haciendo para atender la petición.

Las interrupciones juegan un papel fundamental, en especial en la operación de dispositivos E/S, ya que les permite enviar estas peticiones al cpu. Sin ellas el sistema debería verificar constantemente los dispositivos para comprobar su actividad, pero las interrupciones permiten que los dispositivos puedan permanecer en silencio hasta el momento que requieren atención del procesador. Estas peticiones pueden ser generadas no solo por dispositivos hardware, sino también

por los programas, e incluso en circunstancias especiales (errores generalmente) por el propio procesador.

3.3.1 Funcionamiento

Cuando un dispositivo reclama atención del procesador es para que haga algo. Este "algo" es lo que se conoce como servicio; controlador o gestor de la interrupción, ISR (Interrupt Service Routine). En cualquier caso se trata siempre de ejecutar un programa situado en algún lugar de la memoria RAM o en la ROM. Ocurre que las direcciones de inicio de estos programas, que se conocen como vectores de interrupción, se copian en una tabla de 1024 bytes que se carga al principio de la memoria de usuario (direcciones 0000h a 03FFh) durante el proceso de inicio del sistema, razón por la cual estas rutinas se conocen también como servicios del BIOS.

Esta tabla es llamada tabla de vectores de interrupción (IDT, Interrupt Description Table) y en sus 1024 bytes pueden almacenarse 256 vectores de 4 bytes, es decir, los vectores de interrupción son punteros de 32 bits a las direcciones donde comienza cada rutina. Estas direcciones se guardan en forma segmentada.

Al diseñar el 8088, Intel estableció un reparto de estos vectores, reservando los 5 primeros para uso interno del procesador (para atender las excepciones); a continuación estableció otros 27 de uso reservado, aunque no señaló ningún uso específico para algunos de

ellos. A partir de aquí, los vectores 32 a 255 estaban disponibles según se muestra en la tabla siguiente:

Tabla 3.3.1 La tabla muestra los vectores de interrupción y su descripción.

VECTOR	USO
0	Error: División por cero
1	Excepciones para depuración (ejecución paso a paso)
2	Interrupción no enmascarable
3	Punto de ruptura interrupción (Instrucción INT)
4	Desbordamiento (overflow); utilizado cuando un cálculo aritmético se desborda. Instrucción INTO
5	Reservado
6	Código de instrucción no válido
7	Coprocesador no disponible
8	Fallo doble
9	Desbordamiento del segmento del coprocesador
10	TSS no válido
11	Segmento no disponible
12	Excepción de pila
13	Protección general
14	Fallo de página
15	Reservado
16	Error de coprocesador
17-31	Reservado

32-255	Disponibles para interrupciones enmascarables

Sin embargo, aunque teóricamente las interrupciones 0 a 31 estaban restringidas, IBM y Microsoft utilizaron algunas de ellas sin respetar las indicaciones de Intel. En concreto, IBM y Microsoft utilizaron muchas de ellas para los servicios BIOS. Además de los servicios del BIOS, existen otros, instalados por el sistema operativo, los denominados servicios del sistema.

La forma de trabajar es como sigue: cuando se recibe la petición de interrupción, el microprocesador termina la instrucción que está ejecutando, guarda el contenido de los registros, deshabilita el sistema de interrupciones; ejecuta el servicio y vuelve a su punto de ejecución. El servicio suele terminar con una instrucción IRET (Interrupt Return) que restituye el contenido de los registros y vuelve a habilitar el sistema de interrupciones. En cierto sentido, el proceso es similar al que ocurre cuando aparece la invocación de una función en el código de un programa.

3.4 El proceso de arranque de la computadora

La mayoría de nosotros nos hemos preguntado que pasa en el tiempo que existe desde que se presiona el botón de encendido y el momento en que podemos empezar a dar ordenes a la computadora. Durante dicho lapso la máquina no permanece ociosa, sino todo lo contrario. Ejecuta rutinas de autoprueba, de configuración y de inicialización del entorno de trabajo todo esto para que cuando el

usuario encienda la computadora tenga un ambiente de trabajo familiar y hecho a sus necesidades.

3.4.1 Rutinas que se ejecutan durante el arranque

Cuando se enciende la computadora automáticamente se ejecutan varias rutinas que permiten poner en marcha a la computadora. Estas pruebas corresponden a pequeños programas grabados en una memoria ROM llamada BIOS, cuyas tareas son las siguientes:

- Se encargan de poner en funcionamiento o despertar a la computadora. Para el efecto, al recibir el voltaje de alimentación en el encendido (y por consiguiente, una señal de RESET) el microprocesador busca una instrucción que se encuadra en la localidad 0000h del bus de direcciones, la cual corresponde al inicio del programa de arranque almacenado en el BIOS. Esta rutina le indica al CPU los periféricos que tiene conectados así como la manera de comunicación que va a tener con ellos.

- Comprueba si los elementos de hardware declarados en el sistema están listos para trabajar (rutina POST, entre las que se incluyen la comprobación del CMOS setup, y en las máquinas que así estén configuradas, verificación de la paridad de memoria).

- Permiten al microprocesador mantenerse en comunicación con todos los periféricos.

- Actúan como interfaz entre la máquina y el sistema operativo y, a través de esté, con los programas de aplicación.

Dichas rutinas se encuentran grabadas en uno o dos circuitos de memoria que van alojados en la tarjeta madre, los cuales reciben el nombre de BIOS. Estos circuitos son chips de memoria ROM, aunque para llevar a cabo sus funciones tienen que consultar la información grabada en un pequeño bloque de memoria RAM, la cual por lo general está incorporada en un circuito auxiliar (el reloj de tiempo real).

En el segmento de ROM se almacenan todas las rutinas básicas de comunicación entre los componentes principales de la máquina: microprocesador, memoria, periféricos, entre otros. En el bloque de RAM se graban los datos específicos del hardware de un sistema en particular.

3.4.2 Verificación del hardware

Una vez que se enciende la computadora, se ejecuta un programa de verificación automática del estado general del sistema, llamado POST (Power On Self Test). Entre los elementos de hardware que se revisan durante el arranque, están la misma ROM, el microprocesador, los controladores de interrupciones, los accesos directos a la memoria (DMA´s), el coprocesador matemático (si existe) y

todos los demás elementos en la tarjeta principal. También se revisa la presencia de elementos indispensables para el encendido, como la tarjeta de video, la memoria RAM, las controladoras de disquetes y discos duros.

Cuando finalmente se han comprobado todos los componentes necesarios en la operación del sistema, en la pantalla se despliega un recuadro que indica al usuario que la máquina esta lista para trabajar. Entonces se inicia el proceso de arranque desde el punto de vista del sistema operativo.

Comprobada la integridad del sistema, en la misma rutina grabada en el BIOS aparece una orden para que el microprocesador busque un sistema operativo en el sector de arranque del disco flexible, en el disco duro, en la unidad de CD ó en una unidad de red; esto según el orden de arranque que se tenga en el BIOS.

3.4.3 Carga del sistema operativo

En el momento que concluye la verificación del hardware, el BIOS busca en el sector de arranque de las unidades de disco declaradas una serie de instrucciones denominadas *bootstrap*, que le indican al sistema que busque el archivo con las instrucciones que servirán como complemento a las rutinas básicas de entrada y salida grabadas en el BIOS. El archivo correspondiente se llama IO.SYS (en MS-DOS 5.0 o superior), y debe estar presente en cualquier disco capaz de hacer arrancar la computadora.

A continuación, la máquina busca un segundo archivo, denominado MSDOS.SYS, el cual junto con el anterior constituyen en sí el sistema operativo, donde van contenidas todas las instrucciones para el manejo tanto del hardware como del software que se ejecute sobre él. Estos archivos se encuentran en el directorio raíz de la unidad de arranque, aunque tienen atributos de oculto y de sistema; por eso no se despliegan en pantalla cuando se da la orden DIR. Sin embargo, si se retiran los atributos de sólo lectura, sistema y oculto de estos archivos con el comando ATTRIB –R –S –H *.SYS, al momento de dar la orden DIR aparecerán dichos archivos.

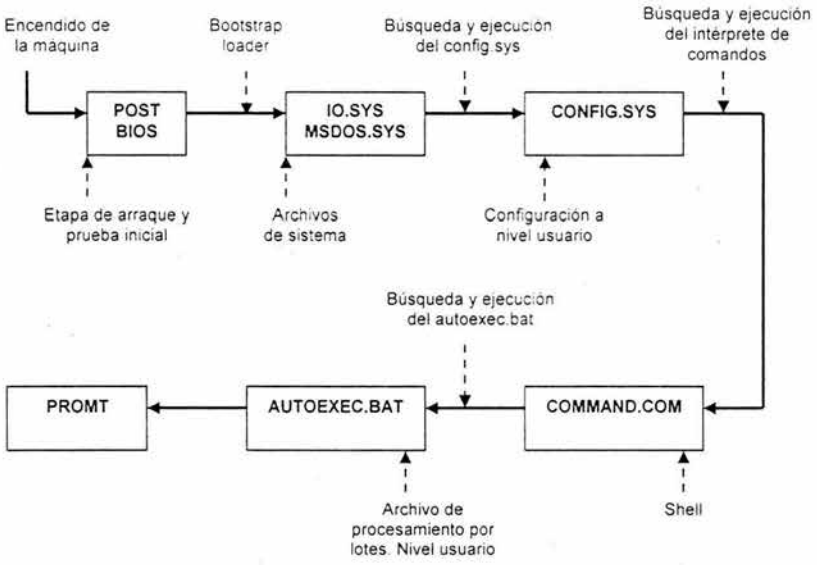


Figura 3.4.3 Diagrama a bloques de la secuencia de arranque de una computadora

El sistema operativo contenido en los archivos IO.SYS y MSDOS.SYS es muy básico, pues es la base de la arquitectura original de la PC (estructura y parámetros válidos en la creación de archivos, estructura de directorios y subdirectorios). Por opción predeterminada se manejan en forma directa dos unidades de disquete, uno o dos discos duros, un teclado, un monitor en modo texto e incluso una impresora. Sin embargo, con la constante aparición de nuevos aditamentos, fue necesario que pudiera personalizarse cada computadora; entonces se indica al sistema operativo que algún elemento no estándar está conectado a la máquina, sin necesidad de tener que modificar los archivos de arranque.

Por ello, una vez que se han leído los archivos IO.SYS y MSDOS.SYS el sistema operativo busca y, en caso de encontrarlo, ejecuta un archivo denominado CONFIG.SYS, cuya función es indicar tanto las particularidades que tendrá el propio sistema operativo, como la existencia de algún elemento externo que se vaya a utilizar de ahí en adelante, como sería un lector de CD, memoria por arriba de 1 MB, entre otras cosas.

3.4.4 El archivo config.sys

Cuando surgió la plataforma PC, los diseñadores de IBM consideraron que un pequeño altavoz interno (*beeper*) sería suficiente para que la máquina emitiera avisos audibles al usuario; estamos hablando de los pitidos y el sonido que se escuchan cada vez que se arranca la máquina. Conforme avanzaron las aplicaciones, en especial

los juegos de computadora. se requirió de un sistema de sonido mejorado; por eso es que se agregó una tarjeta de sonido.

Mas como los elementos no formaban parte de la estructura original de la PC, se tenían que dar de alta en algún punto del arranque, de modo que a partir de ese momento. el sistema supiera que ya tenía incorporado este nuevo periférico y que, por consiguiente, los programas que lo solicitaran tuvieran acceso a él.

Para dar de alta una tarjeta de sonido, es necesario introducir algunas instrucciones en el archivo CONFIG.SYS; lo mismo se puede decir, por ejemplo de la unidad lectora de CD o de algún tipo de escáner. A estas instrucciones se les denomina manejadores (*drivers*); son proporcionados por el fabricante y se configuran automáticamente durante el proceso de instalación (en la mayoría de los casos).

Con el archivo CONFIG.SYS es posible notificar al sistema operativo no solo la presencia del hardware fuera de los estándares originales de la plataforma PC, sino también algunos parámetros que faciliten las tareas cotidianas con la computadora como por ejemplo el número de archivos a mantener abiertos en un momento determinado, la cantidad de memoria reservada para la realización de ciertas tareas, entre otras cosas.

En este archivo también se puede indicar si se desea utilizar un intérprete de comandos distinto al COMMAND.COM, el cual se explica a continuación.

3.4.5 El archivo command.com

Sabemos que una computadora trabaja con números digitales llamados bits, esto es, con 1's y 0's. Cada combinación de 8, 16 ó 32 bits le indican al microprocesador una orden distinta, que puede ser desde una simple lectura de memoria hasta complejas operaciones de multiplicación y transformación de variables. Si el usuario tuviera que aprender todas las órdenes binarias necesarias para el manejo de los diversos programas, la computación personal simple y sencillamente no habría sido posible. Para evitar esta situación, en todos los sistemas operativos modernos se incluye una interfaz cuyo objetivo es servir de intérprete entre una serie de órdenes sencillas impartidas por el usuario, para que las instrucciones complejas binarias sean utilizadas por el microprocesador.

En casi todos los sistemas operativos de disco para PC, el intérprete de comandos recibe el nombre de COMMAND.COM; el cual contiene los comandos internos del DOS, tales como DIR, COPY, TYPE, entre otros. Estos comandos eran muy conocidos por los usuarios del tradicional DOS; pero a la fecha, con la popularidad que tienen los ambientes gráficos de trabajo como Windows, poco a poco están cayendo en el olvido.

Como ejemplo y aprovechando que el CONFIG.SYS se lee antes que el COMMAND.COM; en el primer archivo se puede indicar al sistema operativo que se va a utilizar un *shell* distinto, para que de ahí

en adelante tome en cuanta que las órdenes primarias no deben provenir del COMMAND.COM, sino del intérprete alternativo.

3.4.6 El archivo autoexec.bat

Independientemente del interprete de comandos que se este utilizando, el último archivo que se lee durante el arranque es el AUTOEXEC.BAT. Como su nombre lo indica, es un archivo de proceso por lotes (*batch*) que reúne una serie de órdenes que se desea que el sistema ejecute cada vez que se enciende la máquina.

Por ejemplo, si se tiene un ratón instalado, sería muy conveniente que estuviera disponible cada vez que arranque el sistema. Pues bien, por medio de una orden dada en el AUTOEXEC.BAT, se activa este periférico cada vez que se enciende la PC; lo mismo la disposición del teclado, el entorno del sistema, cualquier programa que se quiera tener residente en memoria (como serían los antivirus), las rutinas que terminarán de dar de alta elementos nuevos de hardware, entre otros.

Una vez ejecutado el AUTOEXEC.BAT, por fin aparece el símbolo de sistema; esto indica que la máquina está lista para comenzar a recibir órdenes. Es decir, a partir de ese momento ya es posible ejecutar los programas.

CAPÍTULO 4. DESCRIPCIÓN DEL PROTOTIPO

Los navegadores o programas de navegación para Internet constituyen el medio de presentación de información para el usuario. Esta información puede presentar características muy variadas en cuanto a su propia naturaleza (texto, imagen, audio, vídeo, etc.), así como en cuanto al soporte requerido para su generación y visualización (tratamiento de textos, hoja de cálculo, tratamiento de imágenes, etc.).

Dada esta variedad de información disponible, los navegadores deben contar con el soporte necesario para ofrecer al usuario la información de la forma adecuada. Este soporte está constituido por un conjunto de programas o aplicaciones complementarias, diseñadas para llevar a cabo actividades especializadas y concretas, específicas para cada tipo de información, los cuales son invocados por los navegadores como complementos a su función de visualización.

En general, la información que puede ser visualizada en Internet sin ayuda de estos complementos se encuentra limitada a páginas creadas mediante el lenguaje HTML (Hyper Text Markup Language), con contenido básico de texto e imágenes de formato específico GIF (Graphics Interchange Format) o JPEG (Join Photographic Experts Group).

La reproducción de una secuencia de sonido o de vídeo, la introducción de elementos interactivos en una página Web como puede ser un fragmento con características de hoja de cálculo, entre otras

cosas, precisa de la colaboración de herramientas externas que permitan extender la funcionalidad de los navegadores.

De ahí que, los navegadores necesiten otras aplicaciones para visualizar los contenidos de Internet, aplicaciones que pueden o no incorporarse como parte de su contenido. Cada nueva versión de navegador incorpora nuevos complementos, facilitando así su disponibilidad inmediata sin necesidad de recurrir a una posterior instalación de los mismos.

4.1 Internet y las páginas Web

Hoy en día, los sistemas de comunicación y las nuevas tecnologías están cambiando gradualmente nuestra forma de relacionarnos y de vivir en sociedad. Esta revolución tiene su eje central en la red Internet. Esta red, a la que se añaden cada día miles de personas, está creando una especie de universo paralelo: una proyección del mundo real en un mundo virtual, en donde la distribución y utilización de información es el pilar fundamental.

La popularización y estandarización de Internet se ha producido gracias a la enorme expansión que ha tenido uno de sus servicios principales: la World Wide Web, la telaraña global de páginas Web enlazadas entre sí.

Las páginas Web comenzaron siendo simples documentos de texto que usuarios de distintas universidades colocaban en la red para

compartir sus proyectos con otras personas situadas en lugares remotos. Al poco tiempo ese sistema resultó de gran utilidad, y poco a poco se fueron añadiendo nuevas funciones que han convertido las páginas Web en lo que conocemos hoy en día: documentos multimedia que pueden contener imágenes, vídeo, sonido, entre otras cosas; capaces de realizar cualquier cosa y desde las que podemos acceder a otras páginas con un simple clic de nuestro ratón.

En muy pocos años hemos asistido a la aparición de sitios Web que ofrecían información casi sobre cualquier tema, hemos utilizado páginas capaces de explorar Internet en busca de alguna página que nos resolviese cualquier duda, a la aparición de programas cada vez más sofisticados para navegar por esas páginas y a la llegada del comercio electrónico y de las grandes multinacionales a la red.

4.2 La aparición de flash

En este proceso de maduración de la red, se ha hecho necesaria la aparición de nuevos sistemas para conseguir más espectacularidad en las páginas Web e interactividad con el usuario. Así han surgido varios leguajes de programación o la modificación de otros para adaptarse a los nuevos tiempos. En este camino hacia una red más versátil apareció la tecnología *Flash*.

En la actualidad, el 99% de los usuarios de Internet podrían estar soportando ya el estándar *SWF*, es decir, tendrían instalada alguna versión del *Player* de *Flash*. El 80% de *Webmasters* y programadores

de páginas *Web* podrían estar utilizando alguna presentación creada por *Macromedia Flash*. Más del 75% de los lectores de guías de informática podrían haber elegido los manuales escritos por y para *Flash*. *Macromedia* puede jactarse de tener el software más popular del momento. No es un sistema operativo, no es un procesador de textos que solamente escribe, no es una aplicación para buscar y bajar música de Internet. *Macromedia Flash* es mucho más que eso. Es un nuevo estándar. Es el futuro de las páginas *Web*. La nueva cara de las aplicaciones multimedia.

La gran aceptación de la tecnología *Flash* por parte del público en general ha disparado la alarma comercial, y otros desarrolladores de software como *Adobe*, han adoptado la tecnología *Flash* para algunos de sus productos, como se demuestra con *Ilustrator 9* y *LiveMotion*, este último en clara competencia con *Flash*.

Evidentemente, Adobe no ha sido el único en adoptar esta solución; otras aplicaciones de desarrolladores de *shareware*, también adoptan o permiten exportar los trabajos en formato SWF, hecho que acrecienta la popularidad de la tecnología de *Flash*. Por otro lado, desde la aparición de *Corel Draw 10* se pueden exportar trabajos bajo el formato de *Flash*. Para conseguirlo, *Macromedia* en colaboración con *Corel*, ha puesto un parche que permite añadir esta opción a *Corel Draw 10*.

Lo que sí es evidente, es que las películas de *Flash* o los contenidos dinámicos de *Flash*, como las llaman algunos *Webmasters*,

podrán ser generadas muy pronto por aplicaciones que no sean específicamente la aplicación de *Flash* de *Macromedia*. Algunos programadores de shareware se esfuerzan, por incluir este estándar en sus aplicaciones, como por ejemplo sucede con *Draw*, *FlashTyper*, *Moho 2.0* o *Swish* por citar algunos. Este último permite crear presentaciones dinámicas basadas en texto y en gráficas, en un entorno casi idéntico a *Macromedia Flash*.

Mediante la tecnología *Flash* podemos diseñar página Web que contienen elementos gráficos, texto, sonido, vídeo y otros, con animaciones y capaces de interactuar con el usuario. Son páginas con mucha animación y movimiento, con textos que aparecen más nítidos, con efectos visuales realmente espectaculares en algunos casos.

Estas son algunas de las ventajas que obtenemos al usar *Flash*:

- Las películas de *Flash* ocupan poco espacio, ya que utilizan distintas técnicas de compresión, con lo que se consigue que se transmitan rápidamente por Internet.

- Las imágenes se pueden almacenar como vectores, por lo que no importa el tamaño en que se visualice la película, las imágenes siempre se mostrarán nítidas y sin defectos. Además, las imágenes vectoriales ocupan menos espacio que los mapas de bits habituales.

- *Flash* dispone de las mismas herramientas que los programas de creación de imágenes más populares, por lo que no necesitamos ningún otro programa para crear nuestras imágenes. Del mismo modo, el tratamiento del color no ofrece límites, permitiendo componer nuestros propios colores basados en tonos planos y degradados.

- Con *Flash* podemos crear animaciones de cualquier elemento de una forma rápida y sencilla. Podemos mover, girar, deformar o cambiar el tamaño.

- Los elementos gráficos sólo se descargan una vez. *Flash* puede crear modelos (llamados símbolos) con los elementos que se repitan para que estos se almacenen más de una vez. Otro sistema para ahorrar espacio.

- Los proyectos creados con *Flash* son muy versátiles, ya que en una misma película podemos insertar todos los elementos necesarios, en lugar de tener cada imagen, sonido, etc., por separado o mantener los elementos más grandes separados de la película principal. De este modo podremos actualizar cualquiera de estos elementos sin necesidad de modificar la película.

- Es un formato protegido. Las películas que colocamos en Internet no pueden ser editadas o modificadas, por lo que el trabajo estará seguro.

- El tratamiento de sonidos también es muy completo. Se puede insertar sonidos en varios formatos y manipularlos mediante *Flash*.

- Las películas de *Flash* muestran los textos tal y como se crearon, incluso si el tipo de letra utilizado no esta disponible en la computadora del usuario, ya que se pueden almacenar las fuentes dentro de la propia película. Éste es un problema muy común en las páginas Web tradicionales y que aquí se encuentra resuelto.

- *Flash* dispone de su propio lenguaje de programación, mediante el que se pueden realizar operaciones numéricas, tratamiento de texto, modificar los elementos de la película mientras ésta es ejecuta. El lenguaje *Actionscrip* amplía de forma ilimitada las posibilidades de *Flash*.

- Con *Flash* podemos tomar imágenes y vídeos creados con otros muchos programas, ya que soporta una gran variedad de formatos gráficos. Además, podemos generar vídeo, archivos GIF animados, secuencia de imágenes, entre otras. Todo ello a partir de una misma película. Incluso podremos crear contenidos para componer CD-ROM interactivos.

En definitiva, *Flash* es la herramienta más popular del momento y el estándar a seguir, en donde el único límite se encuentra en nuestra

imaginación, y es por esto es que se ha decidido desarrollar el prototipo utilizando Flash como herramienta.

4.3 Otras herramientas en el mercado

Algunas de las herramientas que existen actualmente se van a describir aquí, pero hay que señalar que a pesar de que no son iguales al prototipo que vamos a presentar, si tienen que ver con la enseñanza de sistemas operativos, es por ello que se hace pertinente presentarlas. Además es preciso indicar que estas herramientas están disponibles en Internet y no tienen costo alguno. A continuación vamos a hacer una breve descripción del funcionamiento de cada una de estas herramientas.

4.3.1 Minix

Cuando Unix era joven (versión 7), el código fuente se encontraba en todas partes, con autorización de AT&T, y se estudiaba frecuentemente. John Lions, de la Universidad de New South Wales en Australia, llegó a escribir un pequeño folleto que describía su operación, línea por línea. Este folleto se utilizó (con permiso de AT&T) como libro de texto en muchos cursos universitarios sobre sistemas operativos.

Cuando AT&T entregó la versión 7, empezó a comprender que Unix era un valioso producto comercial, así que emitió la versión 7 con una licencia que prohibía el estudio del código fuente en cursos con el objeto de evitar poner en peligro su condición como secreto comercial.

Muchas universidades protestaron simplemente descartando el estudio de Unix y enseñando sólo teoría.

Por desgracia, el solo enseñar teoría dejaba al estudiante con una visión desproporcionada de lo que en realidad es un sistema operativo. Los temas teóricos que por lo general se comprenden con lujo de detalle en cursos y libros de sistemas operativos, como los algoritmos de planificación, en la práctica no son realmente tan importantes. Los temas que en realidad son importantes, como la E/S, procesos y los sistemas de archivo, por lo general se desprecian porque existe muy poca teoría acerca de ellos.

Para remediar esta situación, el conocido y respetado computólogo, Andrew Tannebaum, decidió escribir un nuevo sistema operativo de la nada que sería compatible con Unix, desde el punto de vista del usuario, pero completamente diferente en el interior. Sin siquiera utilizar una sola línea de código de AT&T, este sistema evadía las restricciones de la prohibición, de modo que podía utilizarse para dar una clase o para el estudio individual. El nombre Minix surgió de mini-Unix porque era lo suficientemente pequeño, para que alguien que no fuera un maestro en sistemas operativos pudiese entender la forma en que trabajaba.

Además de la ventaja de eliminar los problemas legales, Minix tenía otra ventaja sobre Unix. Aquél se escribió una década después de Unix y se estructuro en forma más modular. El sistema de archivos de Minix, por ejemplo, no es parte del sistema operativo en absoluto, y

corre como un programa de usuario. Otra diferencia es que Unix se diseñó para ser eficiente; Minix se diseñó para ser legible (en vista de que se puede hablar de que cualquier programa de 12649 líneas es legible). El código de Minix, por ejemplo, tenía más de 3000 comentarios en él.

Minix se diseño para ser compatible con la versión 7 (V7) de Unix. La versión 7 se utilizó como modelo debido a su simplicidad y elegancia. A veces de dice que la versión 7 no solo fue una mejora sobre todos sus antecesores, sino que también lo es sobre todos sus sucesores.

Al igual que Unix, Minix se escribió en lenguaje de programación C y tenia como objetivo ser fácil de portar a diversas computadoras. El desarrollo inicial se hizo en la IBM PC, puesto que esta computadora tenía un uso extenso.

Apegándose a la filosofía de "lo pequeño es bello", Minix no requería de un disco duro para correr, con lo cual se ajustaba a los presupuestos de muchos estudiantes. Minix al estar disponible para cualquier estudiante de ciencias de la computación del mundo, pronto creo una legión de seguidores, incluyendo sus propios grupos de noticias de USENET.

Para el usuario promedio que se sentaba frente a una IBM PC, la ejecución de Minix era muy similar a utilizar cualquier sistema Unix. Muchos de los programas usuales en Unix, como cat, grep, ls, make y

el shell estan presentes y realizan las mismas funciones que sus contrapartes de Unix. Al igual que el sistema operativo mismo, todos estos programas de servicio se rescribieron completamente de la nada por parte del autor, sus alumnos y por algunas otras personas.

4.3.2 Nachos

El Nachos es un pequeño sistema operativo escrito en C++ que corre sobre una máquina virtual MIPS y que puede ser ejecutado en plataformas Unix de muchos tipos. Fue creado por profesores de la Universidad de Berkeley como soporte a las prácticas de la asignatura de Sistemas Operativos. Nachos contiene lo esencial de un sistema operativo; con él se puede experimentar y diseñar un sistema multitarea, memoria virtual, sistema de ficheros, incluso controlar una red simulada de máquinas. Este sistema operativo es de uso público, el *software* se puede descargar por la red.

El código de Nachos utiliza un subconjunto reducido de C++, básicamente soporta los tipos abstractos de datos (clases). No emplea herencia, polimorfismo, excepciones, ni plantillas, así que una persona con conocimientos de C puede aprender en poco tiempo los rudimentos de C++ necesarios para entender las fuentes de Nachos.

El Nachos trabaja sobre una emulación de una máquina con procesador MIPS. Esta emulación incluye una consola, disco, sistema de interrupciones, arquitectura paginada y controlador de red. El Nachos en realidad se ejecuta como un proceso Unix. De esta forma se

puede desarrollar software y depurarlo con absoluto control, además de que pueden estar ejecutándose varios Nachos en la misma máquina real. Otra ventaja de la arquitectura emulada es que los dispositivos, como no son reales, se pueden hacer muy sencillos de entender y manejar.

4.3.3 Simulador de algoritmos de planificación de procesos

Es un pequeño programa cuyo fin es ayudar a entender el funcionamiento de algunos algoritmos de planificación de procesos. Permite calcular algunos parámetros de ejecución de estos algoritmos y visualizar en un diagrama de Gantt la evolución temporal de los procesos. El simulador funciona de la siguiente manera:

Primero tenemos que introducir los datos necesarios para realizar la simulación en una tabla. En ella: las columnas representan los procesos y las filas los datos a introducir para cada proceso. A excepción del tiempo de ejecución, los demás valores son opcionales. No es necesario utilizar todas las columnas. Para editar el valor de una celda basta con posicionarse en ella y teclear el numero deseado.

Una vez introducidos los datos se pueden ir escogiendo los distintos algoritmos para realizar la simulación con solo hacer click en el cuadro de selección algoritmo. En el caso del algoritmo Round Robin (RR) se puede alterar el valor del *cuanto* actuando sobre el control situado debajo de este cuadro.

Una vez seleccionado un algoritmo; al pulsar en el botón actualizar se muestra automáticamente el diagrama de Gantt. Si se realizan cambios en la tabla de procesos, el grafico desaparece temporalmente hasta que el usuario realiza alguna de las acciones anteriores.

EL diagrama consta de una primera barra donde se muestra el estado del CPU y debajo de ella una barra por cada proceso, mostrando la evolución del estado del mismo. El diagrama permite seguir la evolución temporal de los procesos mediante una barra vertical que puede desplazarse a voluntad. En la parte derecha del grafico se muestra el estado de la pila y el tiempo transcurrido desde el inicio.

4.3.4 Simulador de sistemas operativos

El objetivo de este simulador es implementar de una forma cómoda y rápida la mayoría de los conceptos de los sistemas operativos modernos. El simulador es una librería que puede ser incorporada a un programa para simular un sistema operativo concreto.

El simulador permite reproducir los sucesos de creación, destrucción y conmutación de procesos junto al acceso a memoria. La reproducción de un sistema concreto se realiza implementando una serie de funciones que el simulador llama de forma periódica dependiendo de los sucesos ocurridos en el sistema.

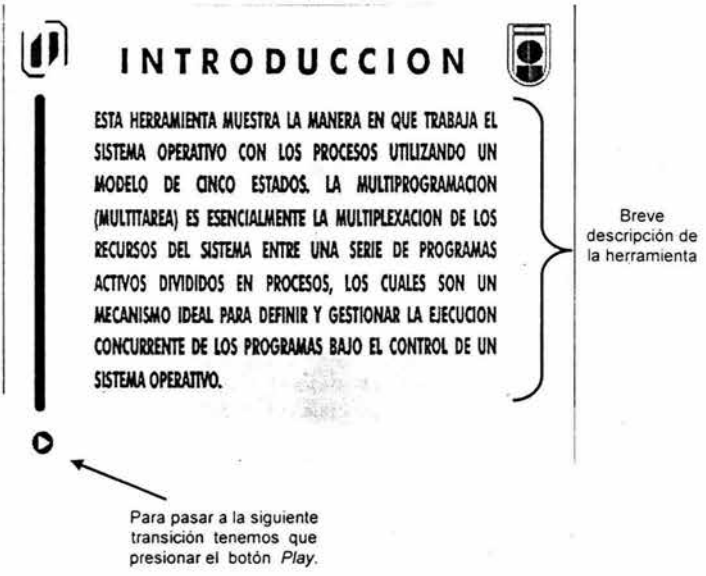
El simulador esta hecho bajo una plataforma de Unix y utilizando C como lenguaje de programación.

4.4 El prototipo

Lo primero que aparece al ejecutar el prototipo es la presentación, que muestra lo siguiente:

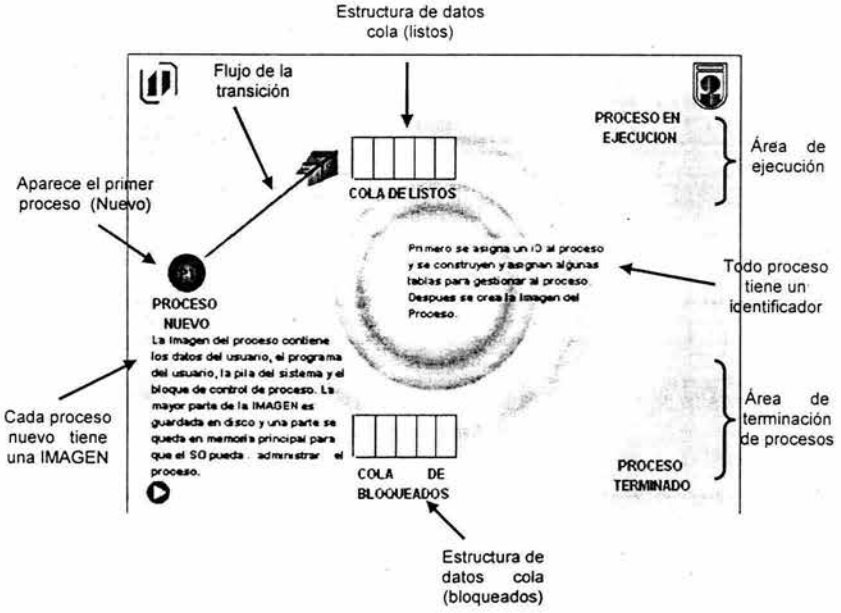
Hay que señalar que en el resto del prototipo aparecerán los logos de la Facultad de Ingeniería y el de la Universidad Americana de Acapulco, los cuales estarán ubicados en la misma posición.

En la siguiente transición, que se da presionando el botón *Play*, nos muestra la una introducción la cual nos da una explicación de lo que trata el prototipo.



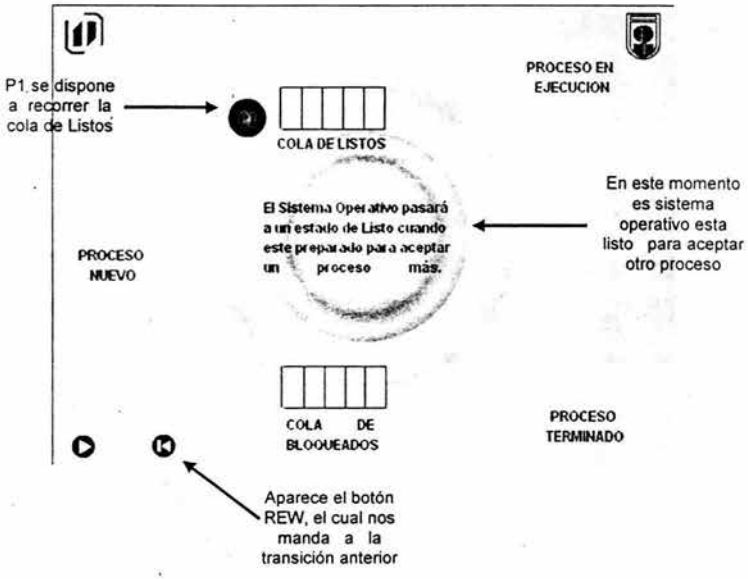
Al momento de presionar *Play* aparece el modelo de cinco estados sobre el cual se va a trabajar en el resto del prototipo.

Los cinco estados son los siguientes: Nuevo, Listos, en Ejecución, Bloqueados y Terminados; y sobre los cuales se van a realizar todas las combinaciones y transiciones que existen para este modelo.



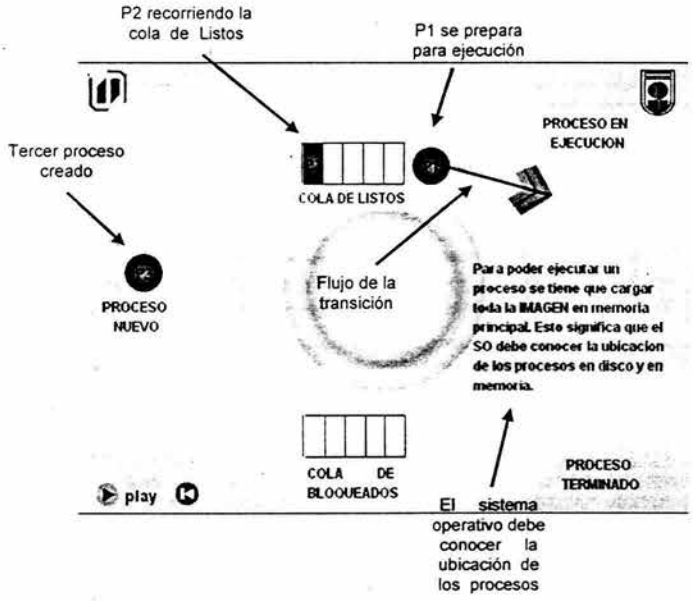
Durante cada transición la herramienta va describiendo los sucesos más importantes que le pasan a un proceso desde que es creado hasta que se destruye.

Al presionar *Play*, el proceso uno (P1) avanza del estado Nuevo a la cola de Listos y se dispone a recórrela.



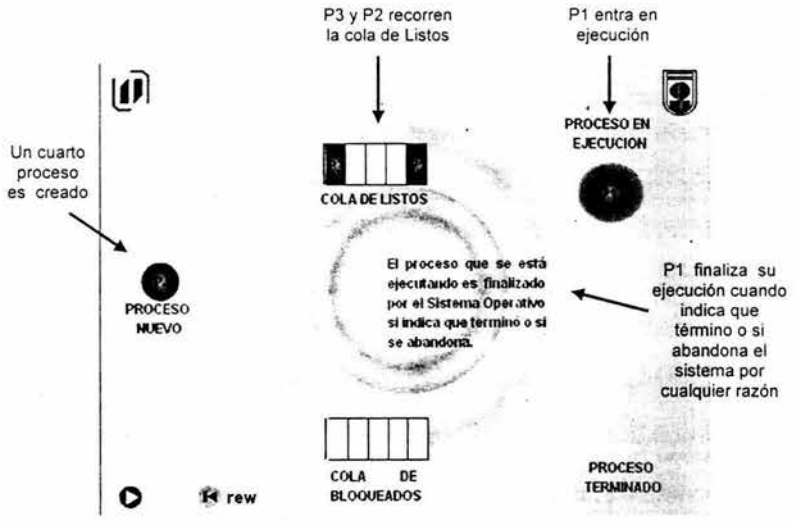
Hasta este momento el resto de los estados no tienen labor alguna.

P1 recorre la cola de Listos y se dispone a ser ejecutado, para lo cual se debe cargar la IMAGEN en memoria. Además es creado un segundo proceso nombrado P2 el cual empieza a recorrer la cola de Listos.



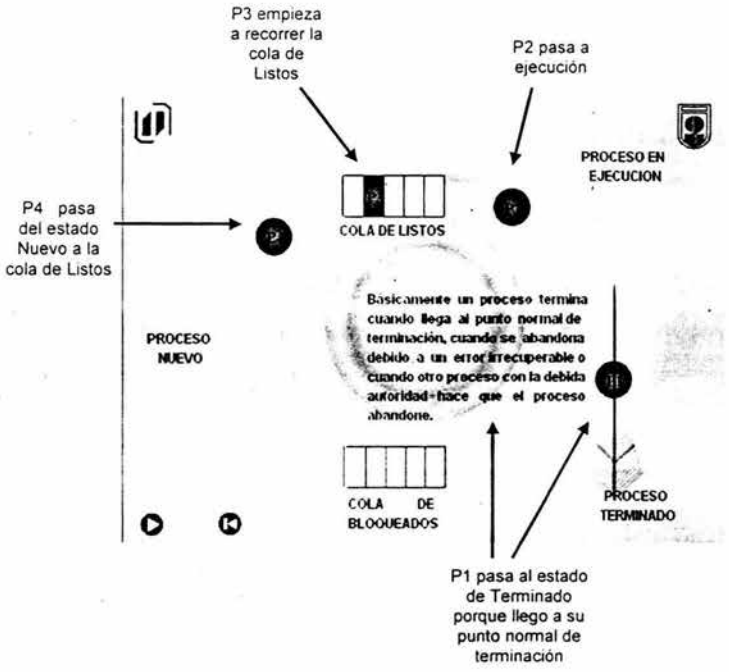
También aparece en escena un tercer proceso (P3).

En este momento P1 entra en ejecución. Los procesos P2 y P3 recorren la cola de Listos y esperan su turno para ser ejecutados. En ese momento es creado un cuarto proceso.



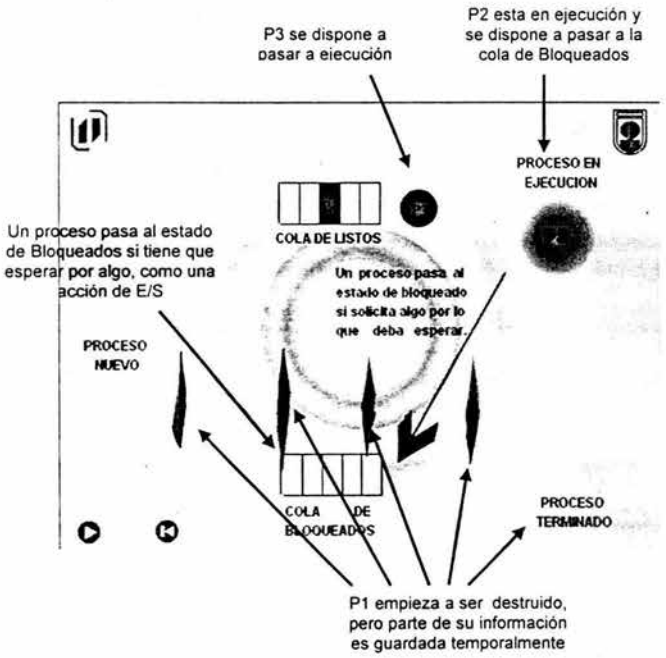
Ahora, nuestra herramienta consta de cuatro procesos activos, los cuales son suficientes para mostrar todas las combinaciones posibles del modelo.

P1 esta por concluir el ciclo de vida más sencillo de un proceso: Nuevo → Listos → Ejecución → Terminado.



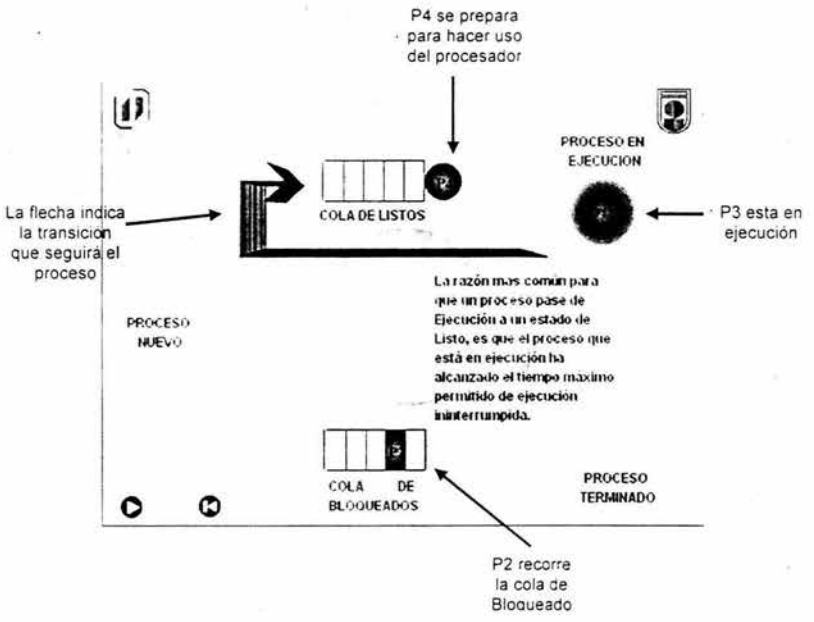
Al momento de que un proceso termina, parte de la información (Imagen del proceso, pilas, entre otras cosas) relacionada con él, es guardada temporalmente con efectos de administración del sistema operativo.

En este instante P2 esta haciendo uso del procesador y el flujo de la transición nos indica que pasará al estado de Bloqueados, esto se debe principalmente a que el proceso debe esperar por alguna acción de E/S, que puede ser una impresión.



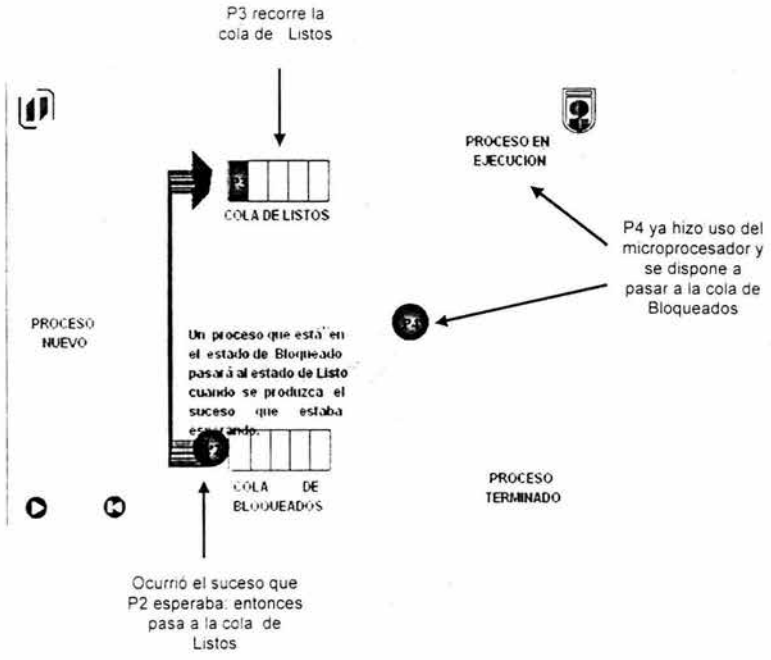
Como podemos observar, la multiplexción es muy clara aquí. El sistema operativo a tiende a todos sus procesos de forma tan rápida que da la impresión que lo hace al mismo tiempo.

En este momento P2 empieza a recorrer la cola de Bloqueados con el fin de esperar a que ocurra el suceso que solicito (Por ejemplo una solicitud de impresión). Así mismo P3 esta en ejecución y esta por pasar a la cola de Listos debido a que esta por terminar su tiempo de ocupación del microprocesador.



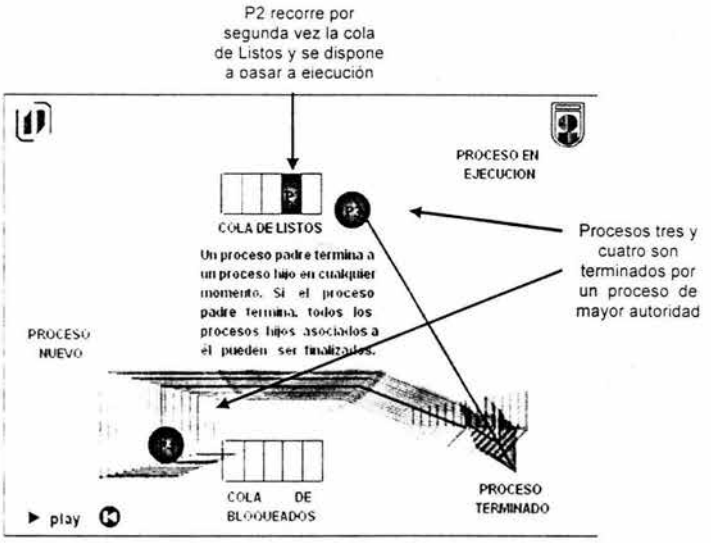
Hay que recalcar que todos los procesos tienen solamente un tiempo limite de uso del microprocesador, esta es una estrategia de planificación, ya que habría procesos que se adueñarían del microprocesador durante un largo tiempo impidiendo que otros lo utilicen y terminen sus tareas.

En este momento P3 ya esta en la cola de Listos debido a que termino su tiempo de uso del microprocesador. De la misma manera, ocurrió la E/S que estaba esperando P2 y ahora pasara a la cola de Listos para esperar turno para ocupar el microprocesador.



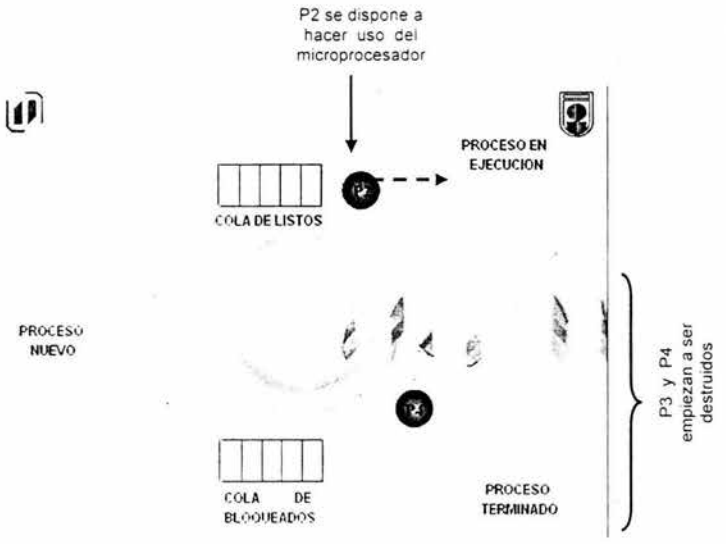
Hay que señalar que P4 ya hizo uso del microprocesador.

En la siguiente transición los procesos dos y tres recorren la cola de Listos y, de la misma manera, P4 recorre la cola de Bloqueados.



En esta ocasión los proceso tres y cuatro son terminados por un proceso con mayor prioridad (proceso padre), el cual tiene toda la autoridad de terminar con sus hijos.

En el transcurso de esta transición los procesos tres y cuatro empiezan a ser destruidos.



P2 pasa a ejecución, hace uso del procesador y enseguida termina su ciclo de vida en el sistema.

Enseguida aparece la transición que nos indica que la herramienta ha terminado.

Por último, debido a que el prototipo aquí presentado esta hecho para ser publicado en Internet, es necesario que hagamos una pequeña reflexión acerca de esto.

Con toda la publicidad que se maneja en Internet, podría parecer que todos nosotros tenemos un ancho de banda bastante grande para manejar cantidades enormes de información, pero la realidad es otra. La verdad es que la mayoría de nosotros contamos con una conexión de 56 Kb (4.7 Kb/seg), que es la de nuestro modem, con pequeñas variaciones. Tomando este hecho como base, al desarrollar esta

película tomamos esto en cuenta, es por ello que la base de descarga de la película es de 56 Kb, aunque hay algunos factores externos que podría influir en el proceso de descarga de una película, entre estos factores podemos mencionar el tráfico en Internet, la velocidad de nuestra conexión, una línea telefónica sucia entre otros.

Fuera de esto, la película esta diseñada para evitar al máximo los cuellos de botella y que el usuario no se desespere al momento de estar bajando la película de Internet y se vaya o haga otra cosa.

A continuación vamos a mostrar de manera gráfica esto. Cuando probamos una película, Flash nos proporciona una herramienta nombrada *visor de anchos de banda* con la cual podemos ver la forma como se cargaría la película realmente. La gráfica siguiente muestra este hecho.

- La sección de Película, ofrece detalles acerca del número de fotogramas de la película, el tamaño del archivo, la velocidad de fotogramas y cuánto durará la carga de la película.

- En Configuración, muestra la velocidad de descarga seleccionada, en este caso son 56 Kb. pero se puede modificar esta velocidad utilizando el menú *Depurar* para seleccionar la velocidad de descarga que se quiera probar.

- El Estado muestra el progreso actual de la descarga a la velocidad seleccionada.

A la derecha puede verse el diagrama de barras que muestra la cantidad de datos que deben cargarse para completar cada fotograma. La línea roja tiene como etiqueta 400 B, representa 400 bytes por fotograma, que es la cantidad de datos que se puede cargar durante el tiempo asignado a cada fotograma, a la velocidad de 12 fotogramas por segundo y utilizando un modem de 56 Kb. Mientras el diagrama de barras esté a la altura de esta línea o por debajo de ella, el reproductor de Flash será capaz de continuar la reproducción sin ninguna pausa; si observamos el diagrama podemos darnos cuenta que solamente al inicio de la película habrá una pausa, pero esta es despreciable si vemos el resto del diagrama porque el resto de la barras están por de bajo de la línea roja.

CONCLUSIÓN Y RECOMENDACIONES

El sólo enseñar teoría deja al estudiante con una visión desproporcionada de lo que en realidad es un sistema operativo. Hay varios temas que se describen con lujo de detalle en los libros de sistemas operativos pero que en la práctica no son realmente importantes. Existen temas que realmente son importantes para el estudiante como el manejo de E/S y los sistemas de archivos de los cuales existe poca información. En este trabajo nos enfocamos a otra parte que tampoco es estudiada con profundidad, los procesos, de la cual se desarrolló un prototipo que resume el manejo de los procesos, en este caso se manejaron cinco procesos, exactamente los mismos que maneja Linux.

Es por esta razón que se realizó este trabajo de investigación, tratando de dar no solamente la teoría, sino también la práctica. Tal y como lo hizo el profesor Andrew S. Tannenbaum cuando de la nada (decir "de la nada" es solamente eso, ya que tuvo como base la versión seis de Unix) decidió escribir un sistema operativo al cual llamó Minix (mini-Unix) con el objetivo de apoyar a la enseñanza de los sistemas operativos. De esta manera, los estudiantes de esta época pudieron analizar minuciosamente la estructura interna de un sistema operativo real, tal y como lo haría un estudiante de biología al disecar una rana.

Podemos pensar que las herramientas de enseñanza de sistemas operativos tienen un costo, pero no es así, la verdad es que la mayoría están en Internet y son totalmente gratuitas lo único que

tenemos que hacer es acceder a ellas. Esto es un apoyo para los estudiantes que tienen el entusiasmo de aprender acerca de sistemas operativos.

Con esto podemos decir que nuestra hipótesis es válida, ya que ahora la Universidad Americana de Acapulco cuenta con una herramienta que ayuda en la enseñanza de sistemas operativos y que junto con la teoría en los libros e Internet harán una mancuerna que dará la solidez de conocimientos necesaria que los estudiantes de la comunidad guerrerense necesita.

Por lo anterior, podemos darnos cuenta que en verdad es necesario practicar, y no solo en sistemas operativos, en cualquier campo que estemos hablando, tan solo hay que recordar que *la práctica hace al maestro*.

GLOSARIO

ISA. Arquitectura del Conjunto de Instrucciones.

FMS. Monitor del Sistema de Fortran, es uno de los primeros sistemas operativos que aparecieron.

CPU. Unidad Central de Proceso.

Enlazador. Es un archivo que toma los archivos de código objeto, generado en los primeros pasos del proceso de compilación, y los convierte en un archivo ejecutable o en una librería.

Compilador. Un compilador acepta los programas escritos en un lenguaje de alto nivel y los traduce a otro lenguaje, generando un programa equivalente independiente que puede ejecutarse tantas veces como se quiera. Este proceso de traducción se conoce como compilación.

Interprete. El código fuente es procesado por un programa llamado intérprete. El intérprete también traduce a código máquina, pero sentencia a sentencia. Para cada instrucción o sentencia del lenguaje de alto nivel, el intérprete la traduce a código máquina, ejecuta el código traducido y pasa a la siguiente sentencia. Cada vez que se quiera ejecutar el programa, necesitaremos tanto el código fuente como el programa intérprete.

Firmware. Concepto intermedio entre el hardware y el software. En las computadoras actuales existen una serie de microprogramas contenidos en una memoria ROM que sirven de ayuda a la unidad de control de la computadora. A estos programas permanentemente almacenados en la memoria de sólo lectura se les denomina firmware.

Monolítico. Un sistema operativo de este tipo no tiene una estructura clara y bien definida. Todos sus componentes se encuentran integrados en un único programa.

Microkernel. Es el tipo de sistema operativo más reciente. Su estructura esta más definida, ya que cuenta con diferentes módulos, los cuales tienen funciones especificas.

Multitarea. Es la capacidad para realizar muchas tareas simultáneamente. Por buena lógica, todas las tareas deberían realizarse exactamente a un mismo tiempo, pero al disponer de un único procesador las tareas no se solapan, sino que se ejecutan de forma secuencial, pero eso sí, en fracciones tan pequeñas de tiempo que da una apariencia de que se están realizando al mismo tiempo.

API (Application Programming Interface). Interfaz para la programación de aplicaciones. Es en lo que se basan lo programadores para hacer compatible un programa con el sistema operativo. DirectX, por ejemplo, es un conjunto de APIs creada por Microsoft para mejorar el funcionamiento de aplicaciones multimedia y juegos en Windows 95/98/2000.

Bus. Medio a través del cual se envía y recibe información entre los diferentes dispositivos de una computadora. La velocidad de esta información depende del ancho de banda del bus.

Ancho de Banda. Cantidad de información que se puede enviar en una determinada fracción de tiempo.

FAT (File Allocation Table). Indica en que lugar se encuentra un archivo. Existen dos tipos FAT 16 Y FAT 32.

FTP (File Transfert Protocol). Protocolo para la transferencia de archivos a través de Internet.

USENET. Conjunto de los servidores que permiten el intercambio de comentarios por parte de personas con los mismos intereses en foros de discusión llamados newsgroups.

ARPANET. Red de telecomunicaciones precursora de Internet.

Macro. Es una especie de rutina que puede ser ejecutada en respuesta a algún evento que sucede en el sistema.

Shell. Interprete de comandos. Proporciona una interfaz que permite al usuario comunicarse con la computadora.

Trap, trampa o excepción. Es un evento que ocurre durante la ejecución de un programa que interrumpe el flujo normal de las sentencias.

Drivers o controladores. Son los encargados de actuar como interfaz entre el sistema operativo y los dispositivos de E/S, así es como todos los dispositivos se entienden y trabajan conjuntamente.

GNU. El proyecto GNU fue iniciado en 1984 con el propósito de desarrollar un sistema operativo compatible con Unix y que fuera software libre. GNU es un acrónimo recursivo para "GNU no es Unix" y se pronuncia fonéticamente.

FSF (Free Software Fundation). La fundación de software libre esta dedicada a eliminar las restricciones sobre el copiado, redistribución, entendimiento y modificación de programas de computadora, promocionando el desarrollo y uso de software libre en todas las áreas de la computación.

Threads ó procesos ligeros. Mediante el uso de threads se consigue ejecutar varios procesos en paralelo, de forma que cuando uno de ellos esté esperando algún evento, permita que el microprocesador ejecute alguno de los otros threads que estén en espera.

KDE. Es un potente entorno de escritorio de código abierto para las etaciones de trabajo Unix.

GNOME. Surgió como un esfuerzo por crear un entorno de escritorio completamente libre para sistemas libres. Su objetivo es proporcionar una serie de aplicaciones amigables y un escritorio fácil de utilizar.

LIFO (Last In First Out). Lo último que entra es lo primero que sale. Es el algoritmo utilizado en estructuras de datos para implementar pilas.

Bit (Binary Digit). Digito binario. Es la unidad mínima de información que maneja una computadora, representa un uno ó un cero.

HALT. Este tipo de instrucciones, indican al procesador que deje de ejecutar instrucciones, dejándolo en un estado latente o de espera.

BIBLIOGRAFÍA

- TOKHEIM, ROGER L.
  Principios Digitales
  Tercera Edición
  McGraw-Hill
- GOOKIN, DAN
  Advanced MS-DOS Batch File Programming
  Windcrest Books
- ECKEL, BRUCE
  Piensa en Java
  Segunda Edición, 2002
  Pearson Educación, S.A. Madrid, España.
- DE LA CRUZ LAZO, CESAR RENE
  Fundamentos de Diseño Digital. Con Aplicación a la Arquitectura de Computadoras.
  Editorial Trillas. Octubre, 1988.
- SANCHEZ PRIETO, SEBASTIAN
  Unix y Linux Guía Práctica.
  Alfaomega grupo Editor. Madrid, España.
- MARQUEZ GARCIA, FRANCISCO MANUEL
  Unix. Programación Avanzada.
  Segunda Edición. 2001.
  Alfaomega Grupo Editor.
- MEGHABGHAB, GEORGE
  Introducción a Unix.
  Prentice Hall Hispanpamericana, S.A. 1988.

- MANSOOR SARWAR, SYED
  KORETSKY, ROBERT
  AQEEL SARWAR, SYED
  El Libro de Unix.
  Pearson Educación. Madrid. España. 2002.
- DEITEL, HARVEY M.
  Introducción a los Sistemas Operativos
  Segunda Edición, México, 1999.
  Addison Wesley Longman.
- STALLINGS, WILLIAM
  Sistemas Operativos
  Segunda Edición. Madrid, España, 1997.
  Prentice Hall.
- PASCUAL FRANCISCO.
  Macromedia Flash MX. Guía de Campo.
  Alfaomega Grupo Editor. Madrid, España, 2003.
- UNDERDAHL, BRIAN.
  Macromedia Flash MX. Manual de Referencia.
  McGraw-Hill. España, 2002.
- FLYNN, IDA M.
  McHOES, ANN McIVER
  Sistemas Operativos.
  Tercera Edición. México, 2001.
  International Thomson Editores.
- CARRETERO PEREZ, JESUS.
  ANASAGASTI, PEDRO DE MIGUEL.
  CARBALLEIRA, FELIX GARCIA.

COSTOYA, FERNANDO PEREZ.
Sistemas Operativos. Una Visión Aplicada.
McGraw-Hill. Madrid, España. 2001.

- CARRETERO PEREZ, JESUS.
  CARBALLEIRA, FELIX GARCIA.
  COSTOYA, FERNANDO PEREZ.
  Prácticas de Sistemas Operativos. De la Base al Diseño.
  McGraw-Hill. Madrid, España. 2001.
- TOKHEIM, ROGER L.
  Fundamentos de los Microprocesadores
  Segunda Edición. México, 1985.
  McGraw-Hill
- SILBERSCHATZ, ABRAHAM.
  GALVIN, PETER BAER.
  Sistemas Operativos.
  Quinta Edición. México, 1999.
  Addison Wesley Longman de México, S.A. de C.V.
- MILENKOVIC, MILAN.
  Sistemas Operativos. Conceptos y Diseño.
  Segunda Edición. México, 1995.
  McGraw-Hill.
- Andrew S. TANENBAUM, ALBERT B. WOODHULL.
  Sistemas Operativos: Diseño e Implementación.
  Segunda Edición, Prentice-Hall, 1998.

- CARD , REMY.
  ERIC DUMAS.
  FRANCK MEVEL.
  Programación Linux 2.0: Api del Sistema y Funcionamiento del Núcleo.
  Editorial Gestión 2000.
- J.L. Hennessy.
  D.A. Patterson.
  Computer Architecture: A Quantitative Approach
  Segunda Edición, 1996.
  Morgan Kaufmann Publishers
- D. Culler.
  J.P. Singh.
  A. Gupta.
  Parallel Computer Architecture: A Hardware / Software Approach
  Morgan Kaufmann Publishers,1998.
- Association for Computing Machinery.
  http://www.acm.org
- ACM readings.
  http://portal.acm.org/portal.cfm
- Citeseer service.
  http://citeseer.nj.nec.com/cs
- IEEE Computing Division.
  http://www.computer.org
- Sitio del Libro "Operating Systems Concepts".
  http://cs-www.cs.yale.edu/homes/avi/os-book/osc/index.html

- Review of Operating Systems.
  http://tunes.org/Review/OSes.html
- OS News.
  http://www.osnews.com
- OS Resource Center.
  http://www.nondot.org/~sabre/os/articles
- Simulador de Sistemas Operativos.
  http://www.inf-cr.uclm.es/www/cvillarrubia/
- A Road Map Through Nachos.
  http://www.cs.duke.edu/~narten/110/nachos/main/main.html
- El sistema operativo Nachos.
  http://sopa.dis.ulpgc.es/nachos/
- Página original del Nachos.
  http.cs.berkeley.edu/~tea/nachos.
- Historia del sistema operativo Minix.
  http://www.comsto.org/so/minix.htm
- Instalación de Minix.
  http://pantuflo.escet.urjc.es/~sdemingi/papers/minix-howto.html
- Información sobre Minix.
  http://www.dc.uba.ar/people/materias/so/html/body_minix.html
- Lecciones de Minix.
  http://sopa.dis.ulpgc.es/ii-dso/lecminix/lecminix.htm
- El núcleo de Minix.
  http://www.infor.uva.es/~benja/nucleo-1/nucleo-1.html
- Minix Information Sheet.
  http://www.cs.vu.nl/~ast/minix.html

- Sistemas operativos. Servicios de información. http://labsopa.dis.ulpgc.es
- Breve historia de Minix. http://es.tldp.org/Manuales-LuCAS/LIPP2/lipp-2.0-beta-html/node16.html
- Comunicación entre procesos. http://personal.redestb.es/mick/fso/procesos/procesos.htm
- General Nachos Documentation. http://www.cs.washington.edu/homes/tom/nachos/
- Nachos Page. http://hegel.ittc.ukans.edu/projects/nachos/
- Nachos Internals. http://www.cse.ogi.edu/~nachos/nachos/1995/internals.html
- Nachos Source Code. http://people.cs.uchicago.edu/~odonnell/OData/Courses/CS230/NACHOS/reading-code.html
- Nachos. http://hegel.ittc.ukans.edu/projects/nachos/nachos/nachos.html
- Nachos System. http://www.wiley.com/college/silberschatz6e/0471417432/pdf/nachos.pdf
- Nachos/486. http://www.cs.virginia.edu/~bah6f/nachos486/paper/nachos486.html
- Administración de procesos. http://www.monografias.com/trabajos14/administ-procesos/administ-procesos.shtml

- Gestión de procesos y memoria en núcleos Unix. http://gugs.sindominio.net/proyectos/index.pl?Gestion_De_Procesos_Y_Memoria_En_Nucleos_Unix
- Tesis doctoral Off. Un nuevo enfoque en la construcción de sistemas operativos distribuidos. http://plan9.escet.urjc.es/who/nemo/export/thesis/thesis.html
- Sistemas operativos. http://www.tau.org.ar/base/lara.pue.udlap.mx/sistoper/index.html
- Programación de sistemas Unix. Comunicación entre procesos. http://es.tldp.org/Universitarios/ipc-seminario.html
- Entrenamiento básico de Linux. http://www.linux.cu/manual/basico-html/node130.html
- Primera exploración de Linux desde la perspectiva de un usuario. http://structio.sourceforge.net/guias/AA_Linux_colegio/exploracion-uno.html
- Manual de programación en Linux. http://www.linuxlots.com/~barreiro/spain/GURH_v60/node1.html
- Procesos en Unix. http://www.lab.dit.upm.es/~lprs/presentaciones/concurrencia_t8_1p.pdf
- Mejoras en el planificador de procesos de Linux. http://www.smaldone.com.ar/gnulinux/docs/scheduler/scheduler2.6_es.pdf
- Guía de Slackware. http://www.slackware.cl/guia/arreglo/Indice.html

- Gestión de Procesos
  http://studies.ac.upc.es/EUPVG/SIOP/documentacion/prob_tema2.pdf
- Universidad Politécnica de Cataluña. Documentación.
  http://studies.ac.upc.es/EUPVG/SIOP/#documentacion
- Ejecución de procesos en Unix.
  http://labsopa.dis.ulpgc.es/prog_c/PROCES.HTM
- Libro en línea de sistemas operativos.
  http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/SOF.ht
- Tutoriales de sistemas operativos.
  http://www.abcdatos.com/tutoriales/sistemasoperativos/
- Sistemas operativos. Un paseo por la historia.
  http://spisa.act.uji.es/~peralta/os/
- Cursos de sistemas operativos.
  http://www.universitarios.org/cursos/informatica/s_operativos.htm
- Sistemas operativos distribuidos y en tiempo real.
  http://www-gist.det.uvigo.es/~pedro/pub/sodtr/
- ¿Cómo se crean los sistemas operativos?
  http://www.arrakis.es/~jespejo/
- Manuales de sistemas operativos.
  http://www.manualesgratis.com/manuales/cat.asp?id=13
- Concurrencia en sistemas operativos.
  http://www.dc.uba.ar/people/proyinv/cso/tr.html
- Historia de los sistemas operativos.
  http://www.giga.islagrande.cu/articulocompleto.htm

- Sistemas operativos para redes cliente/servidor. http://www.geocities.com/SiliconValley/8195/noscs.html
- Sistemas operativos distribuidos. Material docente. http://polaris.lcc.uma.es/~antonio/Docencia/sod/MaterialDocente.html
- Red Científica. Ciencia, tecnología y pensamiento. http://www.redcientifica.com
- Site de sistemas operativos de la facultad de ciencias exactas de la Universidad de Buenos aires. http://www.dc.uba.ar/people/materias/so/
- Site de sistemas operativos de Microsoft MSDN http://www.microsoft.com/spanish/MSDN/estudiantes/ssoo/default.asp
- Curso básico de Unix. http://iie.fing.edu.uy/~vagonbar/unixbas/tutorial.htm
- Advanced Linux Programming. http://www.advancedlinuxprogramming.com/
- Operating System Concepts. Avi Silberschatz, Peter Galvin y Greg Gane. http://cs-www.cs.yale.edu/homes/avi/os-book/osc/slide-dir/index.html
- Curso de AS/400. http://www.fing.edu.uy/inco/cursos/sistoper/recursosEnlaces/OS400-Fing.pdf
- Sistemas operativos SUN/Solaris. http://www.fing.edu.uy/inco/cursos/sistoper/recursosEnlaces/SISTEMAS%20SUN%20SOLARIS.pdf

- Apple España.
  http://www.apple.es/
- Linux.
  http://www.linux.org/
- Microsoft Ibérica.
  http://www.microsoft.com/spain/
- Solo Windows 95.
  http://ms.ctv.es/SW95/home.html
- Sistemas operativos, la Guerra continua.
  http://www.macuarium.com/actual/especiales/2004/04/21_fintade sun.shtml
- Practicas de sistemas operativos: de la base al diseño.
  http://www.arcos.inf.uc3m.es/~ssoo-va/ssoo-prac/ssoo-prac.html
- Introducción a los sistemas operativos.
  http://ditec.um.es/isoi/
- Sistemas operativos. Recursos para estudiantes.
  http://os-matiu.dreamhost.com/course/view.php?id=3
- Microeconomía, sistemas operativos y software libre.
  http://bulma.net/body.phtml?nIdNoticia=1846
- En el principio solo fue la línea de comandos.
  http://www.ciberpunk.com/basicos/neal_stephenson.html
- Introducción a los sistemas operativos.
  http://g.unsa.edu.ar/linux/linuxgros/edt1.html
- Operating Systems Course Material.
  http://elqui.dcsc.utfsm.cl/apuntes/so/
- Computer Architecture and Operating Systems Group.
  http://www.caos.uab.es/

- Catedra de sistemas operativos de la Universidad Nacional de Tucuman. http://www.herrera.unt.edu.ar/so/default.asp
- Simulador de sistemas operativos. http://www.inf-cr.uclm.es/www/cvillarrubia/
- Fundamentos de sistemas operativos con énfasis en GNU/Linux http://www.canapro.org.co/~wilfred_com/
- Grupo Linux Chihuahua. http://chihuahua.linux.org.mx
- Manifiesto del software libre. http://manifiesto.cofradia.org/index.html
- Una caso de negocio para el estudio de software de fuente abierta. http://old.hispalinux.es/informes/MITRE/InformeKenwood.pdf
- Manifiesto del software libre. http://red.coral.com.mx/ceyusa/manifiesto/
- Departamento de informática de la Universidad de Castilla-La Mancha. http://www.info-ab.uclm.es
- Universidad Autónoma de Madrid. http://www.ii.uam.es
- Departamento de Informática Universidad de Valladolid. http://www.infor.uva.es
- Procesos, hilos y multiprocesamiento. http://equipe.nce.ufrj.br/gonzalo/SO/mod3/sld001.htm
- Universidad Católica de la Santísima Concepción. http://www.ucsc.cl

- Departamento de electrónica de la Universidad Técnica Federico Santa Maria de Chile.
  http://www.elo.utfsm.cl/
- Novell Loader Modules.
  http://polaris.lcc.uma.es/~eat/services/nlm.htm
- Linux Chile.
  http://www.linuxchile.cl
- Conceptos de procesos y programación concurrente.
  http://www.lab.dit.upm.es/~lprs/presentaciones/concurrencia_t1.pdf
- Manual de prácticas de sistemas operativos.
  http://www.redes-linux.com/apuntes/so1/practicas/pr1.pdf
- Universidad de Oviedo.
  http://web.uniovi.es/
- Arquitectura de los Autómatas Programables.
  http://www.disa.bi.ehu.es/spanish/asignaturas/10574/2.00.pdf
- Gestión de la memoria en Minix.
  http://www.infor.uva.es/~benja/mm/mm.html
- Sistema operativo Unix.
  http://www.arrakis.es/~ofafian/unix-inicio.html
- Instituto Tecnológico de Veracruz.
  http://www.itver.edu.mx/
- Apuntes sobre procesos y deadlocks.
  http://www.dlsi.ua.es/~abia/PC/material/apuntes-procesos.pdf
- Universidad Regiomontana.
  http://www.ur.mx

- Gestión de la memoria y memoria virtual.
  http://www.inf.udec.cl/~apuente/archivos/Presentaci%F3n%20Trabajo%202.pdf
- Administración de la memoria.
  http://www.dc.uba.ar/people/materias/so/datos/cap13.pdf
- Hilos y procesos.
  http://www-lsi.ugr.es/~jagomez/sisopi_archivos/2ProcesosHebras.pdf
- Departamento de computación. Universidad de Buenos Aires.
  http://www.dc.uba.ar/people/materias/so/
- Laboratorio integrado de sistemas.
  http://bochica.udea.edu.co
- Estructura y tecnologia de computadoras.
  http://dac.escet.urjc.es/docencia/ETC-Superior/Tema5p2v2.pdf
- TLDP/Es Lucas.
  http://es.tldp.org
- Departamento de informática. Universidad de Jaén.
  http://wwwdi.ujaen.es
- Departamento de arquitectura y tecnología de sistemas informáticos. http://laurel.datsi.fi.upm.es
- Universidad Carlos III de Madrid.
  http://www.arcos.inf.uc3m.es
- Departamento de lenguajes y sistemas informáticos. Universidad de Granada.
  http://www-lsi.ugr.es
- Lecciones de Minix.
  http://sopa.dis.ulpgc.es/ii-dso/lecminix/lecminix.htm

- Centro de supercomputo de Galicia.
  http://www.cesga.es
- Tecnología de la información para el conocimiento.
  http://www.zator.com
- Soporte técnico para computadoras.
  http://www.servicioalpc.com
- Universidad de Granada.
  http://www-etsi2.ugr.es
- Asignatura de sistemas operativos.
  http://atc1.aut.uah.es/~ssootm/Contenidos.htm
- Lenguaje ensamblador para arquitectura Intel
  http://homepage.mac.com/eravila/asmix86.html
- Productos interactivos.
  http://entren.dgsca.unam.mx
- Página principal de freeBSD
  http://www.freeBSD.org
- Página principal de netBSD
  http://www.netBSD.org
- Página principal de OpenBSD
  http://www.openBSD.org
- Página principal Linux
  http://www.linux.org
- Página principal Mac OS X
  http://www.macosx.org