Universidad Autónoma de Querétaro

Facultad de Informática

Opción de Titulación: Libro de Prácticas

Materia: Sistemas Operativos I

Profesor Titular: M. en C. Lilia López Vallejo

Alumno: Ignacio Mendoza Cárdenas
Generación 1994-1998

Contenido

CAPÍTULO 4 LOS SISTEMAS OPERATIVOS COMO PLATAFORMAS ESTRATÉGICAS

Introducción:

Este Libro de Prácticas fue escrito con el fin de crear un material en el cual se sustente el contenido de la materia de Sistemas Operativos I. El objetivo fue crear un documento que contemple los aspectos de mayor importancia sobre los sistemas operativos más utilizados en el mercado de sistemas informáticos.

Un sistema Operativo es la parte lógica más importante de un sistema de cómputo, sin éste, una computadora no puede hacer su trabajo. Existen ya muchos sistemas operativos, algunos de ellos son de uso general y otros realizan funciones muy particulares; sin embargo nos enfocamos a analizar los aspectos de los sistemas operativos de mayor interés en la industria informática, principalmente UNIX y Windows NT. Hablaremos de ellos por ser las plataformas más ampliamente aceptadas, y por supuesto, por sus grandes capacidades de procesamiento y facilidades administrativas. Los sistemas de cómputo actuales cuentan con procesadores poderosos y en general hardware con amplias capacidades que requieren de un software capaz de poder controlar dichas configuraciones. Estos sistemas operativos han demostrado ser capaces de satisfacer las necesidades de los administradores de sistemas, los cuales se han vuelto cada vez más exigentes al ver que la demanda de los recursos y los sistemas de información que crece de manera exponencial día tras día.

Sin embargo antes de comenzar el estudio particular de los sistemas operativos, se proporciona una introducción generalizada, esto con el fin de ofrecer un panorama inicial muy sencillo para aquellos usuarios que aun no han incursionado en el estudio de esta materia. Posteriormente hablaremos de cada sistema en particular y analizaremos los aspectos más relevantes de cada unos de ellos.

Como se organiza este Libro

Capítulo 1 Introducción a los Sistemas Operativos

En este primer capítulo nos introduciremos al concepto de sistemas operativos y analizaremos la importancia que tienen éstos para los sistemas de cómputo, hablaremos de sus aspectos más sobresalientes y ciertas generalidades que los hacen mantener una similitud en cuanto a sus objetivos y funcionamiento. Posteriormente hablaremos del shell como programa de interfaz entre el usuario y la máquina, y de la relación entre el kernel y el shell. Hablaremos también de las categorías de los sistemas operativos, las cuales distinguen a los sistemas por sus capacidades de trabajo con los procesos y los usuarios, de esta manera sabremos distinguir entre los sistemas monotarea, multitarea y sus variantes, monousuario, multiusuario, etc. Eso con el fin de conceptuar de manera clara las capacidades de cada sistema operativo y la función para la cual fue diseñado. Por otro lado, la evolución de los sistemas operativos expuesta por etapas, desde los años cuarenta hasta nuestros días, nos dará un panorama general sobre su desarrollo, desde las primeras

máquinas primitivas que escasamente permitían hacer cálculos matemáticos, pasando por las primeras computadoras personales en los años ochenta, hasta las modernas máquinas de nuestra época, enfocadas a resolver grandes volúmenes de procesamiento y a las redes de computadoras; incluyendo el procesamiento distribuido. Posteriormente veremos una sección dedicada a la estructura de los sistemas operativos, los sistemas monolíticos, sistemas por capas y las máquinas virtuales; esto para darnos una idea de la estructura general interna y su esquema de trabajo. Finalmente hablaremos de los sistemas operativos y las tecnologías de información modernas, pero con un enfoque claro hacia los sistemas distribuidos, ahí analizaremos sus capacidades y las consideraciones sobre estos, tales como su transparencia, confiabilidad, escalabilidad, flexibilidad y algunos otros aspectos relevantes.

Capítulo 2 El Sistema Operativo Windows NT

Este capítulo se dedica exclusivamente al estudio del sistema operativo Windows NT, se analizarán los aspectos más importantes de este sistema que ha cobrado gran fuerza en el mercado de los sistemas y en la industria informática. En primer lugar daremos una breve reseña histórica para conocer un poco sobre las bases en las cuales se fundamento este sistema, quién fue su creador y en donde se desarrolló éste; así mismo analizaremos los objetivos de su diseño, tales como la extensibilidad, facilidad de transporte, fiabilidad y robustez, compatibilidad y rendimiento. Posteriormente hablaremos sobre las causas del porqué Windows NT está proliferando, ahí encontraremos las características más importantes del sistema y las facilidades de su enfoque administrativo, con lo cual podremos determinar muchas de las causas del grande éxito de este sistema operativo. Sus ventajas y desventajas las veremos posteriormente y con ello nos daremos cuenta que NT ofrece muchas facilidades para los administradores, una gran flexibilidad y múltiples capacidades de crecimiento. También trataremos el tema del entorno de trabajo para conocer los aspectos más importantes sobre su funcionamiento, también es importante conocer las nuevas características que se integrarán en las siguientes versiones de este sistema operativo, todo parece indicar que NT será un elemento clave para la estrategia de negocios en la industria. Finalmente estaremos hablando de la administración y algunos detalles para su configuración y manejo.

Capítulo 3 El Sistema Operativo UNIX

En este capítulo nos proponemos dar a conocer la filosofía del sistema operativo UNIX. Como dicha filosofía se fundamenta en las relaciones existentes entre programas, gran parte del espacio de este capítulo está dedicado a explicar cada herramienta , pero para lograr la explotar al máximo el poder de UNIX, se deben combinar programas y usarlos para crear otros más poderosos. Si se quiere emplear bien este sistema operativo y sus componentes, es preciso no sólo saber utilizarlos, sino también saber cómo encajan en el ambiente.

Al irse difundiendo este sistema, ha disminuido el número de usuarios expertos en su aplicación. Muchas veces hemos visto s usuarios expertos obtener soluciones deficientes a un problema o escribir programas que las herramientas existentes realizarían con mucha facilidad. Desde luego que una solución elegante no se logra sin tener un poco de

experiencia y de conocimientos. Confiamos en que este capítulo servirá de herramienta para ayudar al usuario a que aprenda a hacer uso del sistema UNIX de una manera agradable y útil, sin importar si se trata de un usuario experto o un principiante.

Este capítulo está dirigido tanto a usuarios finales, como programadores y administradores del sistema UNIX, con la esperanza de hacer más productivo su trabajo. Los primeros apartados son básicos y no requieren de experiencia en el sistema para poder entenderlos, sin embargo conforme avanzamos, encontraremos ejemplos más complejos, por lo que se recomienda llevar una secuencia, desde comprender las bases y la estructura interna del sistema UNIX.

La organización del capítulo es la siguiente:

La primera parte titulada *Introducción,* comenzaremos a ver lo que es el sistema operativo UNIX, revisaremos un poco la historia para saber de donde proviene UNIX, cuales fueron sus objetivos de diseño y como se conforma su estructura interna; también abarcamos algunas etapas de su evolución y cuales son las ventajas y desventajas de su uso. Aquí se revisan también algunas de la variantes de este sistema operativo y finalmente nos introducimos al estudio del sistema. Así, de tal manera comenzamos con una exposición preeliminar sobre el inicio de sesión, personalización del ambiente de trabajo, configuración de la terminal y configuración del shell. Posteriormente comienza el estudio del sistema de archivos de UNIX. El sistema de archivos es un aspecto esencial en la operación y uso del sistema operativo, de modo que es preciso conocerlo. Por lo tanto se describen los archivos, los directorios, los permisos y los modos de archivos, terminamos con una descripción general de la jerarquía del sistema de archivos y con una explicación de los archivos de dispositivos.

En la siguiente sección *Comandos más comunes y expresiones regulares*, se comienzan a ver los comandos más útiles en UNIX, se clasifican en *comandos para la manipulación de directorios* como cd, mkdir y pwd; así también los *comandos para la manipulación de archivos*, tales como awk, grep y find entre otros. Posteriormente se ven los comandos para trabajo en red, los cuales son muy importantes para el manejo de este sistema operativo, pues UNIX fue creado con el propósito de ser utilizado para trabajo en red. Enseguida entramos en materia administración con algunas utilerías para la administración del sistema y algunos comandos para la administración de impresión.

Finalmente llegamos a una de las secciones más importantes de este capítulo, *La programación en shell*, comenzamos con una descripción de la estructura de los programas en shell, los caracteres especiales y variables que se utilizan dentro de los guiones. Posteriormente analizamos el funcionamiento y manejo del editor de textos vi, el cual es una herramienta poderosa que nos permite una fácil edición de los guiones de shell. Abarcaremos la modificación de atributos para la creación de guiones ejecutables, el paso de argumentos, filtros y condiciones utilizadas para el control de flujo de los programas. Terminamos esta sección con el estudio del control a través de ciclos como *for, while* y *until*.

Posteriormente llegaremos a una sección en la que nos adentramos un poco más en materia de administración, aquí nos enfocaremos a estudiar la *administración de procesos y utilerías del sistema*. Comenzamos con la administración de procesos, tal como listados de procesos, suspensión de procesos y como enviar los procesos a primero y segundo plano. Después comenzamos con el análisis del sistema de archivos remoto y shell remoto. Terminamos revisando la transferencia de archivos utilizando el protocolo de transferencia *ftp* y el manejo de corro electrónico utilizando los programas *mail* y *mailx*.

Finalmente llegamos a la sección titulada *Administración del sistema UINIX*, aquí nos enfocamos a ver aspectos puramente administrativos, tales como la instalación del sistema operativo, tipos de configuración de los equipos, tareas administrativas más comunes, algunos métodos para la selección de passwords o contraseñas, como cargar dispositivos y controladores, configuración del kernel y finalizamos con algunos aspectos sobre la seguridad del sistema.

En la sección *Administración de una red TCP/IP en UNIX*, veremos aspectos administrativos, pero todos ellos enfocados al trabajo en red utilizando la pila de protocolos TCP/IP, tales como la configuración de la red y la configuración del servicio de nombres de dominio DNS; finalizamos con la descripción de algunos de los servicios más importantes de Internet como Archie, gopher y world wide web.

La ultima sección titulada *El sistema operativo Linux*, está dedicada a este sistema operativo. Veremos una descripción general del sistema, algunas ventajas y desventajas de su uso, las diferentes distribuciones y finalizamos con el proceso de instalación del sistema operativo Linux.

Capítulo 4 Los Sistemas Operativos como Plataformas Estratégicas

En este capítulo nos enfocaremos a hacer un análisis sobre los sistemas operativos con más fuerza en el mercado, estudiaremos las funciones básicas que cualquier sistema operativo debe abarcar, posteriormente haremos un análisis sobre las tendencias de los sistemas operativos y las estrategias utilizadas para satisfacer las demandas del mercado. Revisaremos también una descripción general de cada uno de los sistemas operativos modernos y sus actuales ventajas competitivas. Finalizamos con una sección sobre como elegir el sistema operativo adecuado.

Capítulo 1

Introducción a los Sistemas Operativos

1.1 INTRODUCCIÓN

El software de una computadora es el conjunto de instrucciones que ésta emplea para manipular datos. Sin el software, la computadora sería un conjunto de medios sin utilizar. Al cargar los programas en una computadora, la máquina actúa como si recibiera una educación instantánea; de pronto "sabe" cómo debe operar.

El software es un conjunto de programas, documentos, procedimientos, y rutinas asociados con la operación de un sistema de cómputo. Distinguiéndose de los componentes físicos llamados hardware. Comúnmente a los programas de computación se les llama software; el software asegura que el programa o sistema cumpla por completo con sus objetivos, opera con eficiencia, esta adecuadamente documentado, y suficientemente sencillo de operar.

Es simplemente el conjunto de instrucciones individuales que se le proporciona al microprocesador para que pueda procesar los datos y generar los resultados esperados. El hardware por si solo no puede hacer nada, pues es necesario que exista el software, que es el conjunto de instrucciones que hacen funcionar al hardware.

El software se clasifica en 3 diferentes Categorías: Sistemas Operativos, Lenguajes de Programación y software de Aplicación. Solamente nos enfocaremos a hablar sobre el Sistema Operativo por ser este nuestro objetivo principal.

En general, el sistema operativo es el gestor y organizador de todas las actividades que realiza la computadora. Marca las pautas según las cuales se intercambia información entre la memoria central y la externa, y determina las operaciones elementales que puede realizar el procesador. El sistema operativo, debe ser cargado en la memoria central antes que ninguna otra información.

1.2 SISTEMAS OPERATIVOS

1.2.1 Generalidades de los sistemas operativos.

Un Sistema Operativo es en sí mismo un programa de computadora. Sin embargo, es un programa muy especial, quizá el más complejo e importante en una computadora. El sistema operativo despierta y le da vida a la computadora, hace que reconozca a la CPU, la memoria, el teclado, el sistema de vídeo y las unidades de disco.

Además, proporciona la facilidad para que los usuarios se comuniquen con la computadora y sirve de plataforma a partir de la cual se corren programas de aplicación. Cuando se enciende una computadora, lo primero que hace es llevar a cabo un autodiagnóstico

llamado autoprueba de encendido POST, (por sus siglas en ingles Power On Self Test). Durante la POST, la computadora indentifica su memoria, sus discos, su teclado, su sistema de vídeo y cualquier otro dispositivo conectado a ella; lo siguiente que la computadora hace es buscar un sistema operativo para arrancar.

Una vez que la computadora ha puesto en marcha su sistema operativo, mantiene al menos parte de éste en su memoria en todo momento. Mientras la computadora esté encendida, el sistema operativo tiene 4 tareas principales:

1. Proporcionar ya sea una interfaz de línea de comando o una interfaz gráfica al usuario, para que este se pueda comunicar con la computadora.
   Interfaz de línea de comando(LCUI): Se introducen palabras y símbolos desde el teclado de la computadora, ejemplo, el MS-DOS, UNIX en modo comando.
   Interfaz gráfica del Usuario (GUI): Se seleccionan las acciones mediante el uso de un dispositivo "mouse" para pulsar sobre figuras llamadas iconos o seleccionar opciones de los menús.

2. Administrar los dispositivos de hardware en la computadora. Cuando corren los programas, necesitan utilizar la memoria, el monitor, las unidades de disco, los puertos de Entrada/Salida (impresoras, módems, etc). El sistema operativo sirve de intermediario entre los programas y el hardware.

3. Administrar y mantener los sistemas de archivo de disco. Los sistemas operativos agrupan la información dentro de compartimientos lógicos para almacenarlos en el disco. Estos grupos de información son llamados archivos. Los archivos pueden contener instrucciones de programas o información creada por el usuario. Se mantiene una lista de los archivos en un disco, y se proporcionan las herramientas necesarias para organizar y manipular estos archivos.

4. Apoyar a otros programas. Otra de las funciones importantes del sistema operativo es proporcionar servicios a otros programas. Estos servicios son similares a aquellos que el sistema proporciona directamente a los usuarios. Por ejemplo, listar los archivos, grabarlos a disco, eliminar archivos, revisar espacio disponible, etc. Cuando los programadores escriben programas de computadora, incluyen en sus programas instrucciones que solicitan los servicios del sistema operativo. Estas instrucciones son conocidas como "llamadas del sistema"

1.2.2 El kernel y el shell

Las funciones centrales de un sistema operativo son controladas por el núcleo (kernel) mientras que la interfaz del usuario es controlada por el entorno (shell). Por ejemplo, la parte más importante del DOS es un programa con el nombre "COMMAND.COM" Este programa realiza más de una sola función; el kernel, que se mantiene en memoria en todo momento, contiene el código máquina de bajo nivel para manejar la administración de hardware para otros programas que necesitan estos servicios, y actúa también como interprete de comandos para que el usuario pueda interactuar con la máquina [Singhal, 1994].

Las funciones de bajo nivel del sistema operativo y las funciones de interpretación de comandos están separadas, de tal forma que se puede mantener el kernel DOS corriendo y utilizar una interfaz de usuario diferente, esto es exactamente lo que sucede cuando se carga Microsoft Windows, de tal manera que éste toma el lugar del shell, reemplazando la interfaz de línea de comandos con una interfaz gráfica de usuario.

En UNIX sucede algo similar al utilizar "el shell" mediante el cual se controlan los recursos del sistema operativo, además de esta función, el shell tiene muchas otras funciones que hacen de UNIX un entorno potente y flexible. En UNIX el shell se trata de un intérprete de órdenes y dispone de lo necesario para que se ejecuten dichas órdenes; además se puede utilizar el shell como un lenguaje de programación de alto nivel. El shell actúa directamente con el kernel del sistema y sirve de intérprete de las órdenes emitidas por el usuario, ya que el kernel no es capaz de realizar esta función por sí mismo [Lowell, 1990].

Existen muchos shells diferentes en el mercado, sin embargo en la mayoría de los sistemas operativos actuales, las interfaces de usuario se están implementando en forma gráfica para crear un ambiente más amigables para el usuario.

1.2.3 Categorías de sistemas operativos.

1.2.3.1 Monotarea.

Esta clasificación hace referencia a aquellos sistemas operativos que sólo tienen capacidad para atender una única tarea del CPU en un momento determinado. Es imposible realizar varias tareas al mismo tiempo en una misma máquina, por lo que cada tarea debe ejecutarse en forma individual.

1.2.3.2 Multitarea.

El término multitarea se refiere a la capacidad del sistema operativo para correr más de un programa al mismo tiempo.

a) Multitarea cooperativa.

Los programas son escritos de tal manera que periódicamente inspeccionan con el sistema operativo para ver si cualquier otro programa necesita a la CPU, si este es el caso, entonces dejan el control del CPU al siguiente programa, a este método se le llama multitarea cooperativa y es el método utilizado por el sistema operativo de las computadoras de Macintosh y DOS corriendo Windows de Microsoft.

b) Multitarea con asignación de prioridades.

El segundo método es el llamado multitarea con asignación de prioridades. Con este esquema el sistema mantiene una lista de procesos (programas) que están corriendo. Cuando se inicia cada proceso en la lista el sistema le asigna una prioridad. En cualquier momento se puede intervenir y modificar la prioridad de un proceso organizando en forma

efectiva la lista de prioridad, el sistema operativo también mantiene el control de la cantidad de tiempo que utiliza con cualquier proceso antes de ir al siguiente.

Con multitarea de asignación de prioridades el sistema operativo puede sustituir en cualquier momento el proceso que esta corriendo y reasignar el tiempo a una tarea de más prioridad. Unix, OS-2 y Windows NT emplean este tipo de multitarea.

1.2.3.3 Monousuario.

El término monousuario se refiere a aquellos sistemas que no tienen capacidad para trabajar y atender las tareas de más de un usuario a la vez.

1.2.3.4 Multiusuario.

Un sistema operativo multiusuario permite a más de un solo usuario accesar una computadora. Claro que para llevarse esto a cabo, el sistema también debe ser capaz de efectuar multitareas.

UNIX es el Sistema Operativo Multiusuario más utilizado. Debido a que fue originalmente diseñado para correr en una minicomputadora, era multiusuario y multitarea desde su concepción. Actualmente se producen versiones de UNIX para PC tales como SunOS de Sun Microsystems, Esix, IBM, sunsoft, Unixware de Microsoft y Linux el cual no es comercial. Apple también produce una versión de UNIX para la Macintosh llamada A/UX.

1.2.3.5 Multiproceso.

Las computadoras que tienen más de un CPU son llamadas *multiproceso*. Un sistema operativo multiproceso coordina las operaciones de las computadoras multiprocesadoras. Ya que cada CPU en una computadora de multiproceso puede estar ejecutando una instrucción, el otro procesador queda liberado para procesar otras instrucciones simultáneamente. Al usar una computadora con capacidades de multiproceso incrementamos su velocidad de respuesta y proceso. Casi todas las computadoras que tienen capacidad de multiproceso ofrecen grandes ventajas.

Los primeros Sistemas Operativos multiproceso realizaban dos tipos de multiproceso:

a) Multiproceso asimétrico.

Una CPU principal retiene el control global de la computadora, así como el de los otros procesadores. Esto fue un primer paso hacia el multiproceso pero no fue la dirección ideal a seguir ya que la CPU principal podía convertirse en un cuello de botella.

b) Multiproceso simétrico.

En un sistema multiproceso simétrico, no existe una CPU controladora única. La barrera a vencer al implementar el multiproceso simétrico es que los sistemas tienen que ser rediseñados o diseñados desde el principio para trabajar en un ambiente multiproceso.

1.3 Evolución de los sistemas operativos.

Los sistemas operativos y la arquitectura de las computadoras han evolucionado de manera conjunta para facilitar el uso de las computadoras. Al construir y usar nuevos sistemas operativos fue necesario ciertos cambios en la arquitectura de los equipos de cómputo. Por ello, es conveniente echar una mirada a la historia.

1.3.1 Los años cuarenta y los años cincuenta.

Los sistemas operativos han evolucionado durante los últimos 40 años a través de distintas fases o generaciones que coinciden más o menos con las décadas. Las primeras computadoras electrónicas digitales de los años cuarenta no tenían sistema operativo; las máquinas de esa época eran tan primitivas que con frecuencia los programas se introducían bit a bit mediante grupos de interruptores mecánicos. Posteriormente los programas se introducían en la computadora en lenguaje máquina mediante tarjetas perforadas y se desarrollaron los lenguajes ensambladores para acelerar el proceso de la programación.

En General Motors Research Laboratories se implantó el primer sistema operativo a principios de los años cincuenta para una IBM 701 [Deitel, 1993]. Los sistemas de los años cincuenta generalmente ejecutaban sólo una tarea a la vez y simplificaban la transición entre tareas para obtener la máxima utilización del sistema de cómputo. Estos sistemas se denominaron *sistemas de procesamieto por lotes de secuencia única* [Deitel, 1993], ya que los programas y los datos eran proporcionados a la computadora en grupos o lotes.

1.3.2 Los Años Sesenta.

Los sistemas de los años sesenta también eran sistemas de procesamiento por lotes, pero podían aprovechar mejor los recursos de la computadora mediante la ejecución de varias tareas al mismo tiempo. Estos sistemas incluían muchos dispositivos periféricos como, lectores de tarjetas, perforadoras de tarjetas, impresoras, unidades de cinta y unidades de disco, pero era raro que una tarea utilizara con eficiencia todos los recursos de la computadora.

Los diseñadores de sistemas operativos observaron que mientras una tarea esperaba a que se completara una operación de entrada o de salida para poder seguir utilizando el procesador, otra podía aprovechar el procesador ocioso. De manera similar, cuando una tarea estaba utilizando el procesador, otras podían estar utilizando los diversos dispositivos de entrada y salida. La mejor forma de poder utilizar al máximo la computadora parecía ser ejecutar una combinación de diversas tareas, así que los diseñadores de sistemas operativos desarrollaron el concepto de *multiprogramación,* en el cual varias tareas se encuentran al mismo tiempo en la memoria principal; un procesador se conmuta de una tarea a otra según sea necesario para lograr que avancen varias tareas, mientras se mantienen en uso los dispositivos periféricos.

Normalmente los usuarios no estaban presentes en el centro de cómputo mientras se ejecutaban sus tareas. Por lo general, las tareas se enviaban en tarjetas perforadas y en

cintas de computadora, y permanecían en las mesas de entrada hasta que se podían cargar en la computadora para su ejecución. Con frecuencia la tarea de un usuario debía esperar durante horas, e incluso días antes de que se pudiera procesar. El más pequeño error en un programa, incluso la falta de un punto o de una coma arruinaba la tarea, y entonces el usuario debía corregir el error y esperar a procesar nuevamente su trabajo. El desarrollo de programas en tal ambiente era penosamente lento.

En 1964, IBM presentó su familia de computadoras System 360. Las diversas computadoras de la serie 360 fueron diseñadas para ser de hardware compatible, para utilizar el sistema operativo OS/360 y para ofrecer mayor capacidad de cómputo a medida que el usuario empleaba máquinas más complejas de la serie. Con los años, la arquitectura de la serie 360 evolucionó a la 370 y las series más recientes 4300 y 30X0 [Deitel, 1993].

Se desarrollaron sistemas operativos más avanzados para atender al mismo tiempo a varios usuarios interactivos. Los usuarios interactivos se comunican con la computadora mediante terminales que están en línea (es decir, conectadas directamente con ella). Como el usuario está presente e interactúa con la computadora, el sistema de cómputo debe responder a las peticiones del usuario, pues de otra forma se vería afectada la productividad del usuario. Los sistemas de tiempo compartido se desarrollaron para permitir trabajar simultáneamente a un gran número de usuarios interactivos.

Muchos de los sistemas de tiempo compartidos de los años sesenta eran sistemas con múltiples modos de operación, que también realizaban procesamiento por lotes y aplicaciones de tiempo real. Los principales avances en el desarrollo de sistemas de tiempo compartido de esa década incluyen el sistema CTSS desarrollado en el MIT, el sistema TSS desarrollado por IBM, el sistema Multics desarrollado en el MIT como sucesor de CTSS y el sistema CP/CMS desarrollado por el centro científico de IBM en Cambridge [Tanenbaum, 1993]. Aunque dichos sistemas se habían diseñado para realizar una computación interactiva básica individual, su valor real radicaba en que permitían compartir programas y datos, y demostrar el valor de la computación interactiva en los ambientes de desarrollo de programas. El tiempo de retorno, es decir, el tiempo que transcurre desde que se introduce una tarea hasta que se reciben los resultados, se redujo a minutos o incluso a segundos.

La persona que escribía un nuevo programa ya no tenía que esperar horas o días para corregir los errores; el usuario podía introducir un programa, compilarlo, recibir una lista de errores de sintaxis, corregirlos inmediatamente, volver a compilar y continuar el ciclo hasta que el programa estuviera libre de errores sintácticos. Después el programa podía ser ejecutado, depurado, corregido y complementado con reducciones considerables en el tiempo de su desarrollo.

Una prueba decisiva del valor de los sistemas de tiempo compartido como apoyo al desarrollo de programas se presentó cuando el MIT utilizó un sistema llamado CTSS para desarrollar su propio sucesor, MULTICS. Éste se distinguió por ser el primer sistema operativo importante escrito principalmente en un lenguaje de programación de alto nivel, en lugar de estar escrito en lenguaje ensamblador. Los diseñadores de UNIX aprendieron de

la experiencia, y así crearon el lenguaje de alto nivel C específicamente para implantar UNIX.

Los sistemas operativos CTSS, Multics y CP/CMS incorporaron el concepto de almacenamiento virtual, con el cual, entre otros beneficios, los programas podían hacer referencia a cantidades de memoria mucho mayores que la cantidad real de memoria principal disponible en la computadora [Tanenbaum, 1993]. Esto liberó a los usuarios de gran parte del trabajo de administración de memoria y les permitió concentrarse en el desarrollo de las aplicaciones.

1.3.3 Los años setenta.

Los sistemas de los años setenta eran sobre todo sistemas de tiempo compartido con múltiples modos de operación, que permitían realizar aplicaciones de procesamiento por lotes, de tiempo compartido y de tiempo real. Las computadoras personales estaban en la etapa inicial de su desarrollo y fueron impulsadas por el avance inicial y continuo de la tecnología de los microprocesadores. Los sistemas experimentales de tiempo compartido de los años sesenta se transformaron en productos comerciales sólidos en los años setenta. Con ello aumentaron las comunicaciones entre los sistemas de cómputo en los Estados Unidos, así se generalizó el uso de estándares de comunicaciones TCP/IP del Departamento de Defensa, especialmente en los ambientes de cómputo militares y universitarios. Las comunicaciones en las redes de área local se hicieron prácticas y económicas con el estándar Ethernet desarrollado en el Centro de Investigación de Palo Alto Xerox.

Los problemas de seguridad aumentaron debido a los grandes volúmenes de información que pasaban por las vulnerables líneas de comunicación. La criptografía recibió mucha atención; fue necesario codificar los datos propios o privados de forma que, aunque los robaran, careciera de valor para cualquiera que no fuera el destinatario original.

1.3.4 Los años ochenta.

Los años ochenta fue la década de la computadora personal y de la estación de trabajo. La tecnología de los microprocesadores evolucionó hasta el punto en que fue posible construir computadoras de escritorio tan poderosas como las macro computadoras (mainframes) de los años setenta. Cada cual podría tener su propia computadora dedicada que efectuara la mayor parte de sus trabajos, y podía utilizar las facilidades de comunicación para transmitir datos entre sistemas. La computación se distribuyó en los lugares en donde se necesitaba, en vez de llevar a procesar los datos hasta una central de cómputo a gran escala. Los paquetes de software de aplicación, como los programas de hojas electrónicas de cálculo, los procesadores de texto y los programas de bases de datos, contribuyeron a la evolución de la computadora personal. En los años setenta, sólo las grandes organizaciones podían comprar computadoras y utilizar la computación interactiva; en los años ochenta, casi cualquiera podía tener su propio sistema personal, o tener acceso a uno y aprovechar sus ventajas.

La clave fue transferir información entre computadoras en las redes de computadoras. Así proliferaron las aplicaciones de correo electrónico, transferencia de archivos y acceso a

bases de datos remotas. El modelo cliente/servidor se generalizó: los clientes son los usuarios de la red que requieren la realización de diversos servicios; los servidores son los componentes de la red de hardware/software que prestan esos servicios. Los servidores se dedican por lo regular a un tipo de tarea, como la impresión, la realización de funciones gráficas a alta velocidad o acceso a bases de datos, entre otros.

1.3.5 Los años noventa.

Los años noventa, la era de la auténtica computación distribuida, en la cual los cómputos se dividen en subcómputos que pueden ejecutarse en otros procesadores, en computadoras de precesadores múltiples y en redes de computadoras. Las aplicaciones aprovechan los ciclos de procesador que se solían desperdiciar en las redes de computadoras personales y de estaciones de trabajo de los años ochenta; los sub-cómputos se distribuyen de manera que pueden aprovechar al máximo las computadoras de una red distribuida.

Las redes tienen configuraciones dinámicas; es decir, siguen operando aunque se añadan o eliminen dispositivos y software. La computación está destinada a poseer gran capacidad y a ser muy fácil de transportar. En los últimos años se han introducido computadoras de tamaño extremadamente pequeño y por lo tanto fácil de transportar, lo cual facilita el trabajo de millones de personas al poder llevar consigo una computadora y realizar operaciones en cualquier lugar.

Con el desarrollo de los protocolos de comunicación y de las redes, en especial la red mundial Internet, las personas pueden conectar sus computadoras desde cualquier parte del mundo y transmitir datos con una gran confiabilidad sin importar cual sea su arquitectura.

1.3.6 El por qué de la evolución de los sistemas operativos.

Los sistemas operativos han estado evolucionando constantemente desde sus inicios para cubrir las necesidades de procesamiento de información del hombre, es por eso que durante las pasadas décadas los científicos de la informática se han empeñado en evolucionar los sistemas operativos para satisfacer dichas demandas de procesamiento que cada vez son mayores, por eso, desde aquellas máquinas construidas inicialmente con rudimentarios elementos para realizar las funciones básicas del sistema operativo, hasta los modernos sistemas construidos para soportar el procesamiento de grandes cantidades de datos y aquellos utilizados en las computadoras personales.

La computadora personal ha asegurado su papel en casi todo tipo de organizaciones: grandes, pequeñas, lucrativas y no lucrativas, de servicios y de orientación técnica. A través de todo el mundo, la PC ha comprobado, sin lugar a dudas que es una herramienta de productividad personal y de negocios indispensable, dichas máquinas utilizan poderosos sistemas operativos para trabajo individual o en red como son: Linux, Windows NT, Windows 9x, OS/2 etc. e incluso algunas son utilizadas como servidores en algunos lugares. Más aun, la gran mayoría de las organizaciones que actualmente se apoyan en las computadoras personales también han decidido invertir en el hardware, software y capacitación requeridos para convertir sus PCs aisladas en una red.

Con los sistemas operativos actuales, ha sido posible realizar el trabajo que hace algunos años se consideraba casi imposible de lograr. El avance de las tecnologías de software y hardware han dado grandes beneficios y sobre todo, se han desarrollado sistemas operativos de red con los cuales se logra compartir recursos tanto de hardware como software y de esta manera hacer más eficiente la inversión de los recursos de cómputo.

Algunas de las razones principales por las cuales fue necesario la creación de los sistemas operativos de red son las siguientes:

1. Los negocios, empresas e instituciones dedicaban demasiado tiempo a la administración y soporte de aplicaciones individuales por lo que fue necesario que las aplicaciones residieran en una sola máquina llamada servidor, ya sea que el servidor ejecutara dichas aplicaciones o bien que proporcionara recursos a las estaciones de trabajo para que estas realizaran el trabajo.

2. La necesidad de ahorrar dinero al permitir que los usuarios pudieran compartir el hardware costoso como impresoras o dispositivos de almacenamiento. Incrementar la productividad a través del acceso remoto a recursos y a la información ubicada en otro lugar geográfico.

3. La necesidad de evitar la redundancia de datos al crear archivos de datos compartidos, lo cual eliminaría dicha entrada redundante de datos y el almacenamiento individual.

4. Crear una forma de comunicación en la cual a través de un servicio fuera posible transferir grandes cantidades de datos o información.

1.4 ESTRUCTURA DE LOS SISTEMAS OPERATIVOS

1.4.1 Sistemas monolíticos

En los sistemas monolíticos no existe estructura alguna. El sistema operativo se escribe como una colección de procedimientos, cada uno de los cuales llama a los otros cuando lo requiere [Madnick, 1985]. Lo único que está bien definido es la interfaz de estos procedimientos, en términos de parámetros y resultados y cada uno de ellos es libre de llamar a cualquier otro, si éste último proporciona cierto cálculo útil para el primero.

Para construir el programa objeto real del sistema operativo mediante este punto de vista, se compila en forma individual los procedimientos o los archivos que contienen los procedimientos y después se enlazan en un solo archivo objeto. En términos de ocultamiento de información, ésta es prácticamente nula. Cada procedimiento es visible a los demás (en contraste con una estructura con módulos o paquetes, en la que la mayoría de la información es local a un módulo y donde solo los datos señalados de forma expresa pueden ser llamados desde el exterior del módulo).

1.4.2 Sistemas en capas

Otra posibilidad es diseñar el sistema en capas. El primer exponente de esta categoría fue el sistema operativo THE (Technische Hogeschool Eindhoven) desarrollado en Holanda en 1968 por E.W. Dijkstra [Tanenbaum, 1993]. El sistema THE era un sistema de procesamiento por lotes y fue desarrollado para trabajar con una computadora holandesa llamada "Electrológica X8". Dicho sistema operativo tenía la siguiente estructura:

No.	Descripción
5	PROGRAMAS DE USUARIO
4	ADMINISTRACIÓN DE I / O
3	COMUNICACIÓN ENTRE PROCESOS Y CONSOLA
2	ADMINISTRACIÓN DE MEMORIA VIRTUAL
1	PLANIFICACIÓN DE CPU
0	HARDWARE

Tabla 1.1 Estructura en capas.

En el caso de THE, la estructura en capas era más que nada para modularizar el diseño (se compilaba y enlazaba todo en un solo gran archivo). Cada capa se podía considerar un TDA, ya que encapsulaba datos y las operaciones que manipulan esos datos. Una capa podía usar los servicios sólo de capas inferiores. La capa 1 se encarga de asignar el procesador, programando el timer y las interrupciones del hardware. Encima de la capa 1 el sistema consistía sólo de procesos secuenciales, cada uno de los cuales se podía programar sin necesidad de preocuparse de que múltiples procesos estuvieran compartiendo la CPU. La capa 2 administraba la memoria. Encima de esa capa los procesos no tenían que preocuparse de por saber si había páginas de memoria en disco o no; y así sucesivamente trabajaba hasta llegar a la capa 5, correspondiente a los procesos de usuario.

Una generalización más avanzada del concepto de capas se presentó en le sistema MULTICS. En lugar de capas, MULTICS estaba organizado como una serie de anillos concéntricos, siendo los anillos interiores los privilegiados. Cuando un procedimiento de un anillo exterior deseaba llamar a un procedimiento de un anillo interior, debía hacer el equivalente a una llamada al sistema. Aunque todo el sistema operativo era parte del espacio de direcciones de cada proceso del usuario en MULTICS, el hardware posibilitó el diseño de procedimientos individuales de forma protegida contra la lectura, escritura o ejecución [Tanenbaum, 1993].

Mientras que el esquema de capas de THE era en realidad un apoyo al diseño debido a que todas las partes del sistema estaban, en última instancia, ligadas entre si en un solo programa objeto, en MULTICS , el mecanismo de anillos estaba más presente durante el tiempo de ejecución y era reforzado por el hardware.

Las ventajas que ofrecía este esquema eran: modularización, ocultamiento de información, flexibilidad. Cada capa se desarrollaba y depuraba sin importar las otras capas. Nuevas funciones podían añadirse, y la implementación de las existentes podía modificarse, mientras no se cambiara la interfaz.

Sin embargo también existían algunas desventajas: requería cuidadosa planificación para decidir qué cosa debía ir en cada capa. Por ejemplo, la capa que provee acceso al disco debía estar en una capa más baja que la que administraba la memoria virtual, pero encima del planificador de CPU, pues el controlador probablemente se bloquearía para esperar a que una operación de E/S se completara. Sin embargo, en un sistema grande, el planificador podría llegar a necesitar manejar más información de los procesos que la que era capaz de almacenar en su memoria, requiriendo apoyarse en el disco. Otro potencial problema era la eficiencia, pues algunos servicios pasaban por varias capas antes de llevarse a cabo.

1.4.3 Máquinas virtuales.

Un sistema operativo de tiempo compartido cumple dos funciones: (1) multiprogramación, y (2) proveer una máquina extendida, con una interfaz más conveniente para los programas de usuario que la máquina pura.
En la década de los 70s, a IBM se le ocurrió separar completamente estas funciones en el sistema operativo VM/370 [Tanenbaum, 1993]. El núcleo del sistema, llamado "monitor de la máquina virtual" se ejecuta directamente sobre el hardware, y su único objetivo es la multiprogramación, ofreciendo múltiples máquinas virtuales a la capa superior.

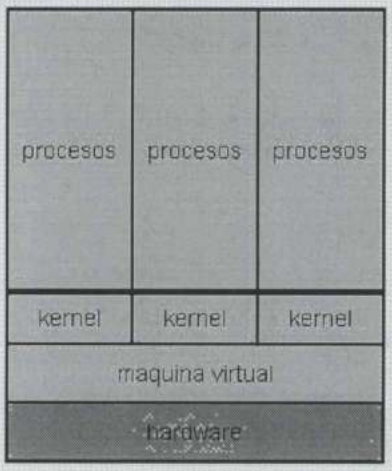

Fig.1.1 Maquina virtual para diferentes kernels de S.O.

Pero estas máquinas virtuales no son máquinas extendidas (con archivos y otras abstracciones), como es lo habitual, sino copias exactas de la máquina original. Gracias a esto, cada máquina virtual puede ejecutar un sistema operativo diferente, construido para operar en la máquina original. La CPU se comparte con planificación de CPU, la memoria con memoria virtual, y el disco se "particiona". Ver la figura 1.1.

No es simple de implementar. La máquina original tiene dos modos: monitor y usuario. El monitor de la máquina virtual opera en modo monitor, y cada máquina virtual puede operar sólo en modo usuario. En consecuencia se debe proveer un modo usuario virtual y un modo monitor virtual. Las acciones que en la máquina real causan el paso de modo usuario a

modo monitor deben causar el paso de modo usuario virtual a modo monitor virtual. ¿Cómo? Cuando un proceso de usuario hace una llamada al sistema, el control se transfiere al monitor de la máquina virtual, quien devuelve el control a la máquina virtual, simulando una llamada al sistema. Si en esas condiciones (operando en modo monitor virtual, que en realidad no es más que modo usuario físico), se intenta accesar el hardware (por ejemplo, una operación de I/O), el control vuelve al monitor de la máquina virtual, quien efectúa, controladamente, la operación).

Este enfoque ofrece ciertas ventajas: facilita el desarrollo de sistemas operativos (cuando la máquina es cara y no se puede entregar una a cada programador). Sin este esquema, es necesario bajar la máquina cada vez que quiere probar una modificación al sistema. También permite resolver problemas de compatibilidad. Por ejemplo, si queremos desarrollar un nuevo sistema operativo, pero queremos que los miles de programas DOS que existen sigan corriendo, podemos usar esta solución, asignando una máquina virtual a cada programa DOS que se ejecute. Se puede ir más lejos, y crear una máquina virtual distinta a la original. Así es como Windows para arquitectura Intel se puede ejecutar, por ejemplo en una Silicon Graphics. El problema es que, en este caso, hay que interpretar cada instrucción en lugar de ejecutarla directamente

1.4.4 Modelo cliente servidor.

Una tendencia de los sistemas operativos actuales es la de mover la mayor parte del código a las capas superiores y eliminar la mayor parte posible del sistema operativo para mantener un núcleo mínimo. El punto de vista usual es el de implantar la mayor parte de las funciones del sistema operativo en los procesos de usuario. Para solicitar un servicio como la lectura de un bloque de cierto archivo, un proceso del usuario (denominado proceso cliente) envía una solicitud a un proceso servidor, que realiza entonces el trabajo y envía la respuesta.

Al separar el sistema operativo en partes, cada una de ellas controla una sección del sistema, como el servicio a archivos, servicio a procesos, servicio a terminales o servicio a la memoria, cada parte es pequeña y controlable. Además, puesto que todos los servidores se ejecutan como procesos en modo usuario y no en modo núcleo, no tienen acceso directo al hardware. En consecuencia, si hay un error en el servidor de archivos, este puede fallar, pero esto no afectará en general a toda la máquina.

Otra de las ventajas del modelo cliente servidor es su capacidad de adaptación para su uso en sistemas distribuidos. Si un cliente se comunica con un servidor a través de mensajes, el cliente no necesita saber si el mensaje se maneja en forma local, en su máquina, o si se envía por medio de una red o un servidor en una máquina remota.

1.5 SISTEMAS OPERATIVOS Y TECNOLOGÍAS DE INFORMACIÓN MODERNAS

1.5.1 Sistemas distribuidos.

Un sistema distribuido es una colección de computadoras conectados por una red de comunicaciones que el usuario percibe como un solo sistema [Singhal, 1994], no necesita saber qué cosas están en qué máquinas ni cómo están conectadas éstas. El usuario accesa los recursos remotos de la misma manera en que accesa a los recursos locales.

En comparación con un sistema centralizado:

- Mejor aprovechamiento de los recursos.
- Mayor poder de cómputo a más bajo costo.
- En teoría, mayor confiablidad, si se maneja suficiente redundancia de datos.
- Crecimiento incremental.

En contraposición:

- El software es mucho más complejo.
- Muchos usuarios desde muchas partes, lo que equivale a problemas de seguridad.

Según nuestra definición, un sistema operativo distribuido debe hacer que los usuarios (procesos) perciban el sistema como un *monoprocesador virtual*. No es necesario que los usuarios estén al tanto de la existencia de múltiples máquinas y múltiples procesadores en el sistema. En la actualidad no hay ningún sistema que cumpla cabalmente con esta definición, pero se realizan avances que finalmente nos llevaran a conformar estos sistemas operativos.

1.5.2 Consideraciones acerca de los sistemas operativos distribuidos.

Los aspectos que hay tener en cuenta en el diseño de un sistema operativo distribuido se describen a continuación.

1.5.2.1 Transparencia.

Los usuarios deben poder accesar los objetos remotos de la misma forma que los accesan los objetos locales. Es responsabilidad del sistema operativo distribuído localizar el recurso y obtener la interacción adecuada.

La transparencia también tiene que ver con la forma de nombrar los objetos: el nombre de un objeto no debe depender del lugar en que se almacena. Un recurso debe poder migrar de un lugar a otro, sin que esto signifique que haya que cambiar su nombre.
Los usuarios, además, deben tener la misma vista del sistema, independientemente del lugar en que el usuario entre al sistema.

1.5.2.2 Confiabilidad.

El hecho de que una máquina se detenga interrumpe todo el sistema, esto puede ser causa de un mal diseño del sistema, de tal manera que la probabilidad de que el sistema esté funcionando en un instante dado debe ser forzosamente del 100% [Singhal, 1994]. Así, si el sistema está hecho de manera tal que cualquier máquina pueda asumir el trabajo de una máquina que se detiene, entonces el sistema será optimo y completamente confiable. Sin embargo la tolerancia a fallas es un tema particularmente complejo, debido a que, en general, no es posible diferenciar entre una falla en el enlace de comunicaciones, una falla en una máquina, una máquina sobrecargada, y pérdida de mensajes.

En la práctica ubicar un punto intermedio razonable, es decir, replicar hasta un punto razonable los elementos claves (datos, servicios, hardware) es lo mejor para convertir al sistema en algo confiable. Obviamente, la confiablidad tiene que ver con la consistencia de los datos. Si un archivo importante se replica, hay que asegurarse que las réplicas se mantengan consistentes; mientras más copias hay, más caro es mantenerlas, y más probable es que haya inconsistencias.

1.5.2.3 Escalabilidad.

La escalabilidad de un sistema es la capacidad para responder a cargas de trabajo crecientes. En particular un sistema distribuido escalable debe diseñarse de manera que opere correcta y eficientemente con diez o con millones de máquinas.

Un principio básico en el diseño de sistemas escalables es que la carga de trabajo de cualquier componente del sistema debe estar acotada por una constante independiente del tamaño del sistema. Si no es así, entonces el crecimiento del sistema estará limitado. En consecuencia, todo lo que se asemeje a centralización, a la larga constituirá un freno a la escalabilidad. Por ejemplo, servidores de autentificación de nombres o de archivos, si éstos son centralizados (uno para todo el sistema), entonces hay un límite en la cantidad de servicios que pueden atender.

Los algoritmos centralizados también deben evitarse. En la medida que sea posible, hay que usar algoritmos descentralizados, que tienen las siguientes características:

- Ninguna máquina tiene el conocimiento completo del estado del sistema.
- Cada nodo toma decisiones basándose únicamente en la información que tiene disponible.
- La falla de una máquina no hace fracasar el algoritmo.
- Los algoritmos no dependen de la existencia de relojes sincronizados, ya que estos pueden fallar.

Todas estas imposiciones son bastante fuertes. No siempre es posible encontrar algoritmos que cumplan todos estos requisitos. Incluso, algoritmos que los cumplen no siempre funcionan bien.

1.5.2.4 Flexibilidad.

Los sistemas distribuidos son relativamente nuevos. Es importante, por ende, que se puedan adaptar a nuevas tecnologías y a nuevos avances.

1.5.3 Sistemas de archivos distribuidos.

Un sistemas de archivos distribuidos o SAD es un sistema de archivos en el que los clientes, servidores y dispositivos de almacenamiento están dispersos en una red, pero la distribución es transparente para los clientes, es decir, el SAD se ve como un sistema de archivos centralizado convencional.

1.5.3.1 Nombres y transparencia.

Los sistemas de archivos (distribuidos y los otros) en general usan simultáneamente dos formas de nombrar archivos: una simbólica, para movimientos del usuario (archivo.ext), y otra con identificadores binarios, para uso interno.

El sistema de archivos debe relacionar (*mapear*) nombres simbólicos con nombres binarios (y para eso se usan los directorios). En un sistema convencional, el nombre binario podría ser simplemente el nodo-i o su equivalente. En un sistema más general como un SAD se requiere que el nombre binario contenga también el identificador de la máquina y del dispositivo dentro de esa máquina. Si además se usa replicación, entonces dado un nombre simbólico el sistema debe obtener no uno, sino un conjunto de nombres binarios: uno por cada réplica del archivo.

1.5.3.2 Patrones de uso de archivos.

- Muchos archivos tienen un tiempo de vida muy corto (por ejemplo los archivos temporales). Sería razonable tratar de crear los archivos en el cliente.
- Pocos archivos son compartidos.
- Patrones de uso según tipos de archivos. Por ejemplo, ejecutables se usan masivamente, pero rara vez cambian.

1.5.3.3 Uso de caché

En un SAD hay cuatro posibles lugares donde poner los archivos o parte de ellos que se están utilizando: el disco del servidor, la memoria principal del servidor, el disco del cliente (si tiene), o la memoria principal del cliente. Lo más directo es manejarlos en el disco del servidor. Si hay espacio, es accesible por los clientes, y al haber una sola copia no hay problemas de consistencia, pero hay problemas de rendimiento y potenciales problemas de congestión; cada acceso por parte del cliente requiere acceso al disco del servidor y transferencia de los datos por la red. Los accesos al disco se pueden reducir sustancialmente si el servidor usa su memoria como caché.

La transferencia al caché puede ser en un bloque o archivos enteros. Como siempre, también hay que definir una política de reemplazo de archivos en caché. Esta opción

también es fácil, y transparente para el cliente. El propio servidor se encarga de mantener sincronizada la información de los archivos que están en memoria con la que está en el disco. Sin embargo, esta alternativa no elimina ni reduce la transferencia de información por la red. La única forma de reducir el tráfico por la red es haciendo *caching* en el cliente, pero ahí es donde empiezan los problemas (independientemente de si se usa la memoria o el disco como caché), porque ahora sí que puede haber inconsistencias.

¿Qué pasa si dos clientes leen el mismo archivo y lo modifican? Si un tercer cliente también lo lee, va a obtener la versión antigua, que todavía está en el servidor. o bien cuando los archivos sean escritos nuevamente al servidor, ¿qué modificación debe prevalecer? Una posibilidad es usar caché *write-through*: cada vez que se escribe en un archivo, el cliente envía la modificación al servidor para que actualice su copia, de cualquier manera, igualmente hay problemas, porque esto no garantiza que se vaya a actualizar el caché de otros clientes que contengan el mismo archivo o bloque.

Una solución mejor puede ser informar las modificaciones al servidor solamente cuando se cierra el archivo. Así mismo, sólo al abrir un archivo, si éste está en el caché, verificar con el servidor si es que esa copia es todavía válida. Es cierto que si dos procesos en distintas máquinas modifican simultáneamente un archivo, los cambios del último que lo cierre prevalecerán, y los del otro se perderán. En sistemas centralizados puede pasar algo similar si dos procesos leen un archivo, lo modifican dentro de sus respectivos espacios de direccionamiento y después lo escriben.

1.5.3.4 Replicación de datos.

Los SAD mantienen (o deberían mantener) múltiples copias de cada archivo, para mejorar la confiabilidad del sistema, de tal manera que si se "cae" un servidor, se usa la copia en otra máquina.

El problema es: ¿cómo manejar la modificación de múltiples copias de un mismo archivo?. Una posible respuesta es que haya un servidor principal que recibe todas las solicitudes y las propaga (cuando tiene tiempo) a los otros. Sin embargo esto tiene desventajas como la centralización en el servidor principal porque si se detiene, ¿qué pasaría?. Tal vez la solución debe tomarse después de un serio análisis.

Finalmente y para terminar, solo diremos que los sistemas operativos constituyen un elemento vital para las organizaciones, ya que a través de ellos se logra realizar el proceso de productividad, no importa cual sistema operativo sea, todos finalmente tienen el mismo objetivo que es "proporcionar una interfaz de interacción entre el usuario y la computadora", permitiendo administrar los componentes de los equipos de cómputo para procesar la información de una manera rápida y eficiente para que el usuario pueda tomar ventaja de las capacidades de una computadora.

Capítulo 2

El Sistema Operativo Windows NT

2.1 INTRODUCCIÓN

2.1.1 Historia.

Tras los problemas presentados por el sistema operativo OS/2, en el año 1988 Microsoft encargó a David N. Cutler "Dave", antiguo consultor senior en Digital Equipement Corporation, liderar el proyecto de creación de un nuevo sistema operativo de nueva tecnología (NT, New Technology) para los noventa [Ezzell, 1997]. Después de casi un año de trabajo, Dave Cutler y su grupo de trabajo, identificaron como básicos los siguientes requisitos:

- *Portabilidad.* Los avances hardware suceden rápidamente y en muchos casos de forma impredicible. Escribir Windows NT en un lenguaje fácilmente portable permitirá moverse libremente de una arquitectura de procesador a otra.
- *Multiproceso y facilidad de ampliación.* Las aplicaciones deberían ser capaces de aprovechar la amplia gama de computadoras disponibles actualmente. Con la aparición de computadoras con mas de un procesador, hacer al NT escalable y con multiproceso, posibilitaría que un usuario ejecutase la misma aplicación en computadoras mono o multiprocesador.
- *Procesamiento distribuido.* Windows NT, debido a las posibilidades avanzadas de trabajo en red, estaría dotado de herramientas de trabajo en red, y además, estaría capacitado para distribuir su trabajo a través de múltiples sistemas de cómputo.
- *Conformidad POSIX.* A mediados y finales de los años ochenta, las agencias de gobierno de EE.UU., comenzaron a especificar el POSIX como un estándar para los contratos informáticos con el gobierno [Ezzell, 1997]. Este estándar, promueve que los fabricantes de interfaces de estilo UNIX hagan sus productos compatibles para que los programadores puedan mover fácilmente sus aplicaciones de un sistema a otro.
- *Certificado de seguridad oficial.* Además de la conformidad POSIX, el gobierno de los EE.UU., también establece unas pautas de seguridad informática para los diferentes tipos de aplicaciones gubernamentales. Lograr un certificado de seguridad aprobado por el gobierno, permite al sistema operativo competir contra otros en ese terreno. Por supuesto, muchos de estos requisitos necesarios son características beneficiosas para cualquier sistema multiusuario. Las pautas de seguridad especifican características como la protección de los recursos de un usuario frente a la intromisión de otro usuario, y establecen cuotas de recursos para evitar que un usuario acumule todos los recursos del sistema (como la memoria). El objetivo de seguridad inicial de Windows NT es el llamado nivel Clase C2, definido por el departamento de defensa de EE.UU., como "protección discrecional" y con herramientas de auditoría para poder contabilizar los sujetos y las acciones que ellos inician". Esto significa que el propietario de un recurso tiene derecho quien puede acceder a él, y el sistema operativo puede detectar cuando y

quien accede a los datos. Los niveles de seguridad de los EE.UU., van desde el nivel A (más riguroso) hasta el nivel D (menos riguroso), pasando por B y C, cada uno de los cuales tienen diversos subniveles [Cuser, 1993]. Aunque Windows NT fue escrito inicialmente para soportar el nivel de seguridad C2, las versiones futuras podrán ajustarse a niveles de seguridad mayores.

Con estos requisitos de mercado, el grupo de desarrollo de Windows NT tuvo una misión: La creación del sistema operativo de Microsoft para la década de los noventa.

2.1.2 Objetivos del diseño.

El diseño de Windows NT necesitó de profundas reflexiones. Para que el sistema pudiera cumplir los requisitos preestablecidos del mercado, era crucial que los rasgos complejos como la conformidad POSIX y la seguridad, fueran incorporados desde el comienzo.

Antes de la codificación, los diseñadores del sistema configuraron cuidadosamente un conjunto de objetivos de diseño. Estos facilitaron la toma de miles de decisiones que determinaron la estructura interna de un proyecto de tan grande envergadura. Las metas u objetivos siguientes son los objetivos de diseño de Windows NT:

- *Extensibilidad*. El código tiene que estar escrito para crecer cómodamente y cambiar según evolucionen las necesidades del mercado.
- *Facilidad de transporte*. Según estén dictadas las metas del mercado, el código debe poder moverse fácilmente de un procesador a otro.
- *Fiabilidad y robustez*. El sistema debería protegerse a sí mismo del mal funcionamiento interno y de los fallos externos. Debería comportarse de forma predecible en todo momento, y las aplicaciones no deberían dañar al sistema operativo o a su funcionamiento.
- *Compatibilidad*. Aunque Windows NT debería extender la tecnología existente, su interfaz de usuario y los API (application programming interface, o interfaz de programación de aplicaciones) deberían ser compatibles con los sistemas ya existentes en Microsoft.
- *Rendimiento*. Dentro de las limitaciones impuestas por el resto de los objetivos de diseño, el sistema debería ser lo más rápido posible en cualquier plataforma hardware.

A continuación veremos mas a fondo los objetivos de diseño de Windows NT, y los efectos que éstos han tenido en la forma final del sistema operativo.

2.1.2.1 Extensibilidad.

Invariablemente, los sistemas operativos cambian a lo largo del tiempo. Los cambios se presentan a menudo en forma de nuevas características.

Asegurar la integridad del código de Windows NT respecto a los cambios del sistema operativo, era un objetivo primordial de diseño. Para esto Windows NT comprende un ejecutor privilegiado y un conjunto de servidores no privilegiados llamados subsistemas protegidos. El término "privilegiado" hace referencia a los modos de funcionamiento del

procesador [Wayatt, 1998]. La mayoría de los procesadores poseen un modo de funcionamiento privilegiado (o quizás varios), en el cual un programa puede emplear todas las instrucciones de la máquina y puede acceder a la memoria del sistema, y un modo no privilegiado en el que ciertas instrucciones no están permitidas y no puede acceder a la memoria del sistema. En Windows NT, el modo privilegiado se denomina modo kernel, y el modo no privilegiado se denomina modo usuario (user mode).

Generalmente, un sistema operativo se ejecuta sólo en modo kernel y los programas de aplicación se ejecutan sólo en modo usuario, excepto cuando solicitan servicios del sistema operativo. Sin embargo, el diseño de Windows NT es único, porque sus subsistemas protegidos se ejecutan en modo usuario al igual que las aplicaciones. Esta estructura permite que los subsistemas protegidos puedan ser modificados o que se añadan nuevos subsistemas sin afectar a la integridad del ejecutor.
Además de los subsistemas protegidos, Windows NT incluye numerosas características para asegurar su extensibilidad:

Estructura modular. El ejecutor consta de un juego discreto de componentes individuales que interaccionan entre ellos solamente a través de interfaces funcionales. Pueden añadirse nuevos componentes al ejecutor de forma modular, ejecutándose su función al llamar a las interfaces suministradas por los componentes existentes.
Utilización de objetos para representar los recursos del sistema. Los objetos, tipo de datos abstractos que son manipulados solamente por un conjunto especial de servicios, permiten que los recursos del sistema sean manejados uniformemente. La adición de nuevos objetos no altera objetos ya existentes ni requiere el cambio del código existente.

El sistema de I/O de Windows NT soporta drivers que pueden ir añadiéndose al sistema a medida que se ejecuta. Se pueden soportar nuevos sistemas de ficheros, dispositivos, y redes, escribiendo drivers de dispositivo, de sistemas de ficheros, o de transporte, y cargándolos en el sistema.
El Mecanismo de llamada de procedimiento remoto (RPC, Remote Procedure Call), que permite que una aplicación llame a servicios remotos sin importar su situación dentro de la red. Se pueden añadir nuevos servicios a cualquier máquina de la red, y pueden ponerse inmediatamente a disposición del resto de los integrantes de la red.

2.1.2.2 Transportabilidad.

Este segundo objetivo, está estrechamente relacionado con la extensibilidad. La extensibilidad le permite a un sistema operativo ser fácilmente mejorado, y la portabilidad le permite a todo sistema operativo ser trasladado a otra máquina basada en otro microprocesador o configuración, con la menor modificación posible del código.

Windows NT fue escrito para poder transportarse fácilmente a máquinas que empleasen direcciones lineales de 32 bits y proporcionasen características de manejo de memoria virtual. También puede moverse a otro tipo de máquinas, pero con un costo mayor.

2.1.2.3 Fiabilidad.

La fiabilidad se refiere a dos ideas diferentes pero relacionadas. Primero, un sistema operativo debería ser robusto, respondiendo de forma predecible a las condiciones de error, incluso ante aquellas originadas por fallos de hardware. Segundo, el sistema operativo debería protegerse activamente a sí mismo y a sus usuarios, de un sabotaje accidental o deliberado de los programas de usuario.

El manejo estructurado de excepciones es un método para capturar condiciones de error y responder a ellas uniformemente. Esta es la defensa principal de Windows NT frente a fallos en el software o el hardware. Siempre que sucede un evento anormal, o el sistema operativo o el procesador generan una excepción; el código de manejo de la excepción, existe en todo el sistema, se invoca automáticamente para responder a la condición, asegurando que ningún error no detectado haga estragos en los programas de usuario o en el mismo sistema.

2.1.2.4 Compatibilidad.

Es el cuarto objetivo de diseño de Windows NT, es un sistema complicado. En general, la compatibilidad se refiere a la habilidad de un sistema operativo de ejecutar código escrito para otro sistema operativo, o para versiones antiguas de es sistema operativo. Para Windows NT, el tema de la compatibilidad toma diversas formas.

Definir esta cuestión es un asunto de compatibilidad binaria versus compatibilidad de aplicaciones a nivel de fuentes. Se consigue compatibilidad binaria cuando se puede tomar un programa ejecutable y ejecutarlo con éxito en un sistema operativo diferente. La compatibilidad a nivel de fuentes implica recompilar el programa antes de ejecutarlo en el nuevo sistema.

El que el nuevo sistema operativo sea compatible en código binario o en código fuente con otro sistema existente, depende de varios factores. El primero de ellos es la arquitectura del nuevo procesador del sistema. Si el procesador emplea el mismo juego de instrucciones y las direcciones de memoria son del mismo tamaño que el antiguo, puede lograrse la compatibilidad binaria.

La compatibilidad binaria no es tan sencilla entre procesadores basados en arquitecturas diferentes. Cada arquitectura de procesador por lo general conlleva un único lenguaje máquina. Esto significa que la arquitectura cruzada y la compatibilidad binaria solo pueden lograrse si se dispone de un programa emulador para convertir un conjunto de instrucciones de código máquina en otro.

Windows NT admite sistemas de ficheros existentes, incluyendo el sistema de ficheros MS-DOS (FAT), el sistema de ficheros de elevadas prestaciones del OS/2 (HPFS, High-performance file system), el sistema de ficheros de CD-ROM (CDFS, CD-ROM File System), y el nuevo y recuperable sistema de ficheros de Windows NT (NTFS).

2.1.2.5 Rendimiento.

Los siguientes procesos ayudaron a lograr este objetivo:

Cada componente de Windows NT fue diseñado teniendo en cuenta el rendimiento. La comprobación del rendimiento y la modelización se realizaron en las partes críticas del sistema. Las llamadas al sistema, los fallos de página, y otros caminos cruciales, se optimizaron cuidadosamente para asegurar la velocidad de procesamiento más rápida posible.
Los subsistemas protegidos (servidores) que llevan a cabo funciones del sistema operativo se tienen que comunicar frecuentemente entre ellos y con aplicaciones cliente. Para garantizar que estas aplicaciones no entorpezcan el rendimiento del servidor, se incorporó un mecanismo de paso de mensajes de alta velocidad denominado llamada de procedimiento local (LPC, Local procedure call), como parte integral del sistema operativo.
Cada subsistema protegido que proporcina un entorno del sistema operativo (subsistema de entorno) fue cuidadosamente diseñado para maximizar la velocidad de los servicios del sistema utilizados con más frecuencia.
Los componentes cruciales del software de red de Windows NT, fueron integrados en la parte privilegiada del sistema operativo para asegurar la mayor efectividad posible. Aunque estos componentes se encuentran integrados, pueden cargarse y retirarse del sistema de forma dinámica.

2.2 PORQUÉ NT ESTÁ PROLIFERANDO

Windows NT es un sistema operativo que ayuda a organizar la forma de trabajar a diario con la PC y ofrece una manera muy sencilla de trabajar. Las letras NT como ya lo mencionamos, significan "Nueva Tecnología". Windows NT Fue diseñado para uso de compañías grandes, por lo tanto realiza muy bien algunas tareas tales como la protección por contraseñas. Pretende dar la impresión de ser un escritorio, de manera que se encuentre en pantalla todo lo que se necesita, gracias a su interfaz gráfica con iconos de colores y dibujos, Windows NT ha logrado ganarse la confianza de miles de administradores de redes y sistemas al ofrecer una Interfaz 100 % amigable al usuario.

Windows NT además ofrece gran seguridad por medio del acceso por cuentas y contraseñas. Es decir un usuario debe tener su cuenta asignada y una contraseña para poder tener acceso al sistema, de otra manera es imposible acceder a este. Contiene protecciones para directorios, archivos, y periféricos, es decir que todo esto se encuentra con una contraseña para poder ser utilizado. El sistema NTFS es el sistema de archivos propio de Windows NT, el cual está basado en un sistema de transacciones, es decir que tiene la capacidad de almacenar una gran cantidad de operaciones a disco para que en el caso de alguna falla, éste elemento pueda ser usado para la reconstrucción del sistema de archivos del disco. En cuanto a seguridad de la información, estas son algunas de las características importantes que hacen del sistema Windows NT un sistema confiable y dado que uno de los puntos de mayor importancia para un administrador de sistemas es la seguridad absoluta del sistema, encontramos plena satisfacción de muchos administradores en este aspecto.

2.2.1 Sus características.

Sin embargo en gran medida el éxito de NT se debe a que cuenta con muchas características que lo hacen un sistema accesible y poderoso. Dichas características son las siguientes:

- Está basado en variaciones del kernel de Mac y de UNIX.
- La arquitectura del microkernel soporta aplicaciones no diseñadas para Windows NT.
- Operaciones básicas de sistemas y otras capas sobre ella.
- Soporta 5 subsistemas:
  - Windows 32 bits.
  - Windows 16 bits.
  - DOS.
  - POSIX.
  - OS/2.
- Funciona como Cliente – Servidor en un ambiente de red.
- Permite desarrollar servicios de redireccionamiento para LAN Manager de Mips, RISC y Digital Alpha.
- Soporta sistemas de multiproceso.
- Cada aplicación se encuentra ejecutando en un hilo tratado como una caja multiprocesadora.
- Al igual que OS/2 ejecuta aplicaciones con errores de codificación, principalmente al ejecutarse en procesadores 386 y 486.
- Cada aplicación es limitada a un espacio de memoria (Esquema de direccionamiento de 32 bits real).
- Ejecuta aplicaciones de 16 y 32 bits y de otros Sistemas Operativos y para RISC de 64 bits.
- Existen versiones para Laptop.
- Soporta la tecnología Plug-in para sistemas API y sistemas de archivos instalables.
- También cuenta con servicios básicos de redes y APIs para archivos, manejadores de impresión, manejo de mensajes y seguridad directa.
- Aplicaciones para redes digitales que pueden ejecutarse en diferentes plataformas.
- Implanta facilidades para el uso de OSF, DCE y RPCs.
- Para facilitar los puertos de aplicación aísla el kernel del Hardware (Tipo de interfaz para el Sistema Operativo), con lo que se logra la portabilidad o compatibilidad a nivel de código.
- Provee datos, aplicaciones y protección del sistema contra accesos inadvertidos.
- Permite a los usuarios un acceso seguro a más información sin comprometer la seguridad del sistema.
- Conserva las principales características del servidor 3.51 incluso el protocolo nativo NetBEUI, IPX y TCP/IP.
- Soporta hasta 256 usuarios, administración de multidominio y replicación de directorio.
- Nuevas o mejoradas herramientas de administración y perfeccionamiento en la ejecución.
- El servidor NT relacionado con Internet, envía la información con el servidor de Internet IIS, también hace uso del FTP.
- Relaciona nuevos rasgos punto a punto con el protocolo PPTP y TCP/IP.

- Ayuda a consolidar la posición de NT como la plataforma del servidor en escenarios de Internet.
- Adopta el estilo de Unix de servicio de dominio DNS como norma.
- Incluye herramientas basadas en el Web referentes a la administración.
- Permite los siguientes modos de autorización.
  - Por usuario.
  - Autorización de la conexión concurrente.

De esta manera Windows NT presenta muchas características importantes que lo hacen ser un sistema altamente competitivo, de tal manera satisface las demandas de muchos administradores que buscan una manera sencilla y eficiente de realizar el trabajo de administración, pero sin necesidad de introducirse al complejo mundo de sistemas como UNIX. Es así como NT logra ganar día con día mayor popularidad entre los sistemas operativos de mejor aceptación en el mercado.

2.2.2 Un enfoque administrativo.

Por otro lado la administración de sistemas es uno de los temas más de moda actualmente. Aunque la administración de sistemas no es realmente algo nuevo, la idea está en el aire desde que los mainframes dominaban la informática. Pero la administración de sistemas es cada vez más importante en los entornos de Windows NT pues cada vez crece el número de aplicaciones y de bases de datos críticas de negocios que se ejecutan sobre NT, de esta manera las tareas administrativas son mucho más flexibles y la compatibilidad está al alcance de la mano, permitiendo compartir e intercambiar información de una manera sencilla.

Administrar estos recursos significa asegurarse que los usuarios pueden ejecutar aplicaciones, acceder a los datos, enviar correo electrónico y completar las otras funciones relacionadas con la informática que requieren sus tareas. La administración de sistemas implica resolver todas las cuestiones administrativas que rodean el uso de ordenadores: ¿Está un ordenador desactivado? ¿Cómo se ejecutan las aplicaciones? ¿Cómo añadimos usuarios al sistema y les garantizamos privilegios? ¿Tiene problemas la red? ¿Cómo podemos desplegar aplicaciones? ¿Cómo podemos asegurar que los usuarios pueden acceder a las aplicaciones? ¿Qué podemos hacer para optimizar el rendimiento? ¿Cómo recuperamos información si un servidor se desactiva? ¿Cómo sigo el cumplimiento de los acuerdos de las licencias de software?

La lista de preguntas que los administradores de sistemas deben responder continúa y la proliferación de sistemas distribuidos añade más tareas de administración a la lista. No hay un único paquete de software que responda a todas estas preguntas pero las iniciativas de administración de Windows NT se enfocan hacia soluciones prácticas y sencillas y ayudan a los departamentos de informática grandes y heterogéneos a ejecutar aplicaciones de misión crítica sobre Windows NT.

2.3 VENTAJAS Y DESVENTAJAS

Windows NT como cualquier otro sistema operativo, ofrece múltiples ventajas y desventajas, sin embargo para un buen administrador, la primera tarea es evaluar dichas ventajas y desventajas y determinar finalmente si el sistema operativo es óptimo y permite satisfacer las demandas para las cuales es requerido. A continuación mencionamos las ventajas y desventajas de Windows NT.

2.3.1 Ventajas de windows NT.

- La instalación es muy sencilla y no requiere de mucha experiencia.
- Es un sistema multitarea.
- Es un sistema multiusuario.
- Apoya el uso de múltiples procesadores.
- Soporta diferentes arquitecturas.
- Permite el uso de servidores no dedicados.
- Soporta acceso remoto.
- Ofrece mucha seguridad en sesiones remotas.
- Brinda apoyo a Macintosh.
- Apoyo para archivos de DOS y Mac en el servidor.
- El sistema está protegido del acceso ilegal a las aplicaciones en las diferentes configuraciones.
- Ofrece la detección de intrusos.
- Permite cambiar periódicamente las contraseñas.
- Soporta múltiples protocolos.
- Carga automáticamente manejadores en las estaciones de trabajo.
- Trabaja con impresoras de estaciones remotas.
- Soporta múltiples impresoras y asigna prioridades a las colas de impresión.
- Muestra estadísticas de Errores del sistema, Caché, Información del disco duro, Información de manejadores, No. de archivos abiertos, Porcentaje de uso del CPU, Información general del servidor y de las estaciones de trabajo, etc.
- Brinda la posibilidad de asignar diferentes permisos a los diferentes tipos de usuarios.
- Permite realizar diferentes tipos de auditorías, tales como del acceso a archivos, conexión y desconexión, encendido y apagado del sistema, errores del sistema, información de archivos y directorios, etc.
- No permite criptografía de llave pública ni privada.
- No permite realizar algunas tareas en sesiones remotas, como instalación y actualización.

2.3.2 Desventajas de windows NT.

- Tiene ciertas limitaciones por RAM, como; número máximo de archivos abiertos y almacenamiento de disco total.
- Requiere como mínimo 16 Mb en RAM, y procesador Pentium a 133 MHz o superior.
- El usuario no puede limitar la cantidad de espacio en el disco duro.
- No soporta archivos de NFS.
- No ofrece el bloqueo de intrusos.

- No soporta la ejecución de algunas aplicaciones para DOS.

2.4 EL ENTORNO DE TRABAJO

Windows NT incorpora la misma interfaz que ha hecho tan famoso a Windows 9x. La instalación, el botón de Inicio y las barras desplegables recuerdan mucho al sistema operativo que acercó tanto la informática al mercado doméstico. Con esto Microsoft ha conseguido que no existan grandes diferencias de uso y presentación entre sus sistemas operativos, lo cual es una de sus preocupaciones principales.

El aspecto de NT es igual al de Windows 9x y actúa del mismo modo; pero desde el punto de vista técnico ofrece mayor seguridad, ejecuta aplicaciones de 16 bits sin problema alguno, además, la gestión y manipulación de archivos es más rápida, eficiente y potente que la de Windows 9x.

La instalación de NT es muy sencilla y se puede realizar a partir de una versión anterior de Windows o desde el mismo MS-DOS.

En primera instancia, se ofrece la posibilidad de elegir entre un sistema de asignación de archivos tipo FAT, o de tipo NTFS (Propia y exclusiva de NT). Esta última permite la recuperación del sistema de archivos, el uso de medios de almacenamiento extremadamente grandes, nombres de archivos largos y otras características para el subsistema Posix, programación orientada a objetos. Después de la elección de la partición donde se va a realizar el volcado de archivos, Windows NT examina la máquina detectando todo el Hardware instalado.

Una vez concluido el proceso de instalación, se presenta una interfaz gráfica prácticamente idéntica a la utilizada por Windows 9x. Incluye algunas mejoras aparecidas en Microsoft Plus!, como son, temas de escritorio, protectores de pantalla, juegos, etc... Pero las diferencias empiezan a apreciarse cuando se pulsa sobre el clásico botón de Inicio.

En la opción de los programas, aparece un apartado de herramientas administrativas desde el que se puede acceder a diferentes utilidades, las cuales permiten obtener un completo control del sistema. Existen los apartados para administrar discos, el acceso remoto y los usuarios, desde los cuales se pueden configurar las opciones de red, si éstas han sido previamente instaladas.

Existen diferencias muy marcadas entre las utilidades que incluía la versión 3.51, y las que incorpora la versión 4.0. La terminal se ha hecho más potente y vistosa, y en la versión 4.0 se incluye una serie de iconos definidos para poder conectarse directamente con CompuServe, Microsoft BBS, etc.

2.4.1 El nuevo windows NT.

El hecho es que cada vez hay más gente que quiere o necesita algo más que DOS, Windows 3.1 y Windows 95 y se plantea el salto a Windows NT con los menores traumas posibles. Esta situación poco a poco está siendo asimilada por Microsoft. El gigante del software

intensifica cada vez más la promoción de Windows NT entre los pequeños empresarios y los usuarios finales; es decir, entre los usuarios para los que fue diseñado Windows 95. Los argumentos de Microsoft tienen peso. En un sistema Pentium con 32Mb de memoria, NT Workstation funciona cerca del 20% más rápido que Windows 95, tanto si es utilizado por un ingeniero, un programador o un estudiante. Hay muchas y buenas razones para que gran parte de los actuales usuarios de Windows 95 se cambien a Windows NT. Si todavía no se ha producido el cambio es por algunos inconvenientes de este sistema operativo, los cuales Microsoft pretende resolver con la versión 5.0 de NT.

La propuesta de cambiarse ahora desde Windows 95 a NT 4.0 Workstation y así preparar el terreno al futuro NT 5.0 Workstation tiene varias ventajas. La primera ventaja de NT Workstation frente a Windows 9x es que es un sistema operativo más estable y predecible. Esta es la cualidad más importante para quienes la computadora es su principal herramienta de trabajo. Un error, cualquier caída inesperada, significan pérdidas de tiempo y dinero. Pero la estabilidad es también uno de los requisitos cada vez más valorados por muchos usuarios ocasionales que utilizan Windows para entretenerse o hacer trabajos menores o semiprofesionales.

Cada vez hay más gente de todos los niveles que está harta de los colapsos y amnesias de Windows 95. A pesar de sus deficiencias, NT ofrece más estabilidad a cualquier nivel de uso. Aunque algunas veces opera con aplicaciones de extraño comportamiento, es decir son mucho muy inestables, es difícil que la computadora se quede congelada o que se planteen problemas críticos e irreparables de colisión con otras aplicaciones. En caso de existir conflictos, NT permite casi siempre guardar los datos, cerrar las aplicaciones que choquen con la actividad principal y seguir trabajando. Otra ventaja que hace aconsejable el cambio a NT es que el hardware exigido por este sistema operativo es cada vez más asequible a los más modestos presupuestos, incluidos los de usuarios finales que compran una PC para trabajar en casa. Hoy, la mayoría de las máquinas que se venden en el mercado tienen 32 Mb de memoria, un disco duro con más de 1Gb de capacidad y un procesador Pentium de, por lo menos 200 MHz. Esta capacidad normal de las actuales PCs es suficiente para obtener un rendimiento aceptable con NT Workstation, y no exigirá mayores desembolsos cuando llegue otra versión de NT. En una máquina como la que se ha descrito, está más que demostrado que las aplicaciones normales rinden al menos un 25% más con NT Workstation que con Windows 95 o 98.

En NT no corren bien algunas aplicaciones que parecen exclusivas de Windows 9x, pero en la mayoría de los casos, siempre que se trabaje en un sistema con esos 32 Mb de RAM, funcionan sin errores las aplicaciones descritas para la plataforma Win32 y Windows 9x. Por tanto, todas las aplicaciones de 16 bits escritas para Windows 3.1 o adaptadas para Windows 95 funcionan también igual o mejor con Windows NT Workstation, y mantendrán o aumentarán su rendimiento cuando funcionen en NT 5.0. Los únicos programas que no trabajan con NT son los que acceden a bajo nivel y directamente contra el hardware , como ciertos juegos diseñados originariamente para DOS, como Duke Nukem 3D o Quake. Pero ni siquiera en estos caso hay que renunciar a operar en modo óptimo con nuestros programas favoritos, ya que el NT 4.0 y 5.0 admiten configuraciones de arranque dual. Otro tipo de software vedado para NT son los antivirus que no estén diseñados específicamente para esta plataforma. La última ventaja evidente de migrar ahora a NT 4.0

y luego a NT 5. Workstation desde Windows 95 es que su entorno de usuario es similar, se puede empezar a trabajar sin realizar un curo, sin siquiera repasar los manuales.

Las tres desventajas principales de Windows NT 4.0 respecto a Windows 9x son sus mayores exigencias de hardware, su defectuosa configuración automática y su interfaz un poco menos amistosa. Parte de estas carencias se han resuelto con el abaratamiento drástico del hardware necesario, con los dos servicios de revisión que ha experimentado NT desde mediados de 1996 y la actualización de su interfaz gráfica al modo de página web, común a Windows 95 con Internet Explorer, a Windows 98 y a NT 5.0 Workstation. En cualquier caso, NT 5.0 no exige una máquina tan cara como antes, y hereda la interfaz de usuario y el sistema de configuración automática de Windows 9x.

Finalmente el objetivo principal de NT es convertirse en un sistema universal para redes de cualquier envergadura.

2.5 LA ADMINISTRACIÓN DEL SISTEMA

La administración del sistema es una de las tareas más difíciles de realizar en los sistemas operativos de red como lo es Windows NT. Sin embargo NT a su vez ofrece una manera sencilla de realizar estas tareas al proporcionar una interfaz amigable a usuario y un modo de trabajo similar a Window 9.x, es por ello que las tareas administrativas suelen ser sencillas bajo esta plataforma.

La mayor parte de la información del sistema y tareas administrativas se pueden realizar en el panel de control, de tal manera que haremos una revisión general de los elementos más importantes con que éste cuenta y algunos ubicados en el menú. Al panel de control se puede acceder de igual manera como se hace en Window 9.x, se puede acceder haciendo click en el botón inicio, configuración y panel de control.

2.5.1 Fecha y hora.

El cuadro de diálogo Fecha y hora permite modificar la fecha y hora del sistema, así como la zona horaria en la que se haya el sistema. En caso de ser necesario, una utilidad del kit de recursos permite modificar y crear nuevas zonas horarias que complementen a las suministradas con el sistema.
Para cambiar la hora y/o la fecha y/o la zona horaria, simplemente cliquear en la barra deslizante y seleccionar la zona horaria deseada. Para cambiar la fecha u hora, posicionarse con el mouse en el lugar deseado, hacer un click y con el tecleado lo cambiamos.

2.5.2 Configuración regional.

NT, al igual que Windows 9.x, está preparado para soportar diversos idiomas y trabajar en diferentes países, con diferentes esquemas de ordenación alfabética, moneda y formas de escribir la fecha, hora y signos de puntuación en números y monedas. Se puede modificar:

- ***Configuración regional***: Permite seleccionar las características del idioma de cada región.

- ***Número***: Selección del formato para números y el sistema de medida.
- ***Moneda***: Selección de la representación de cantidades monetarias.
- ***Hora***: Selección del formato para representar las horas
- ***Fecha***: Selección del formato para representar las fechas.
- ***Idioma***. Permite seleccionar y añadir diferentes idiomas, con sus correspondientes
- ***Configuraciones de teclado***. También permite habilitar atajos del teclado que intercambian las configuraciones del teclado.

También permite cambiar las configuraciones de inicio y apagado. Esto es en la parte de propiedades del sistema, hacemos un click donde dice startup/shoutdown.

2.5.3 Impresoras.

Normalmente la mayoría de las aplicaciones informáticas necesitan una impresora predeterminada para funcionar. Para instalar una nueva impresora al sistema NT dispone de un sencillo asistente que nos guía en el proceso de instalación. Para acceder a este asistente se utiliza el icono *Agregar impresora* del *panel de control*.

En NT la impresora puede estar instalada en el propio equipo o ser una impresora de red. El caso más sencillo consiste en que la impresora esté conectada a un servidor NT, que la ha compartido. En este caso para conectarnos a la impresora basta seleccionar en el asistente la opción Servidor de impresora de red. Al seleccionar esta opción aparece un cuadro de diálogo con las impresoras compartidas por los equipos. Las impresoras compartidas por los NT aparecen aparte, como elementos separados del equipo que las comparte, haciendo más fácil su localización.

La opción instalar la impresora en el propio equipo conduce a una serie de asistentes que nos permiten configurar y compartir la impresora.

2.5.3.1 Panel de Control, Impresoras, Asistente.

El primer paso consiste en seleccionar el puerto al que está conectada la impresora. Puede ser:

- Un puerto paralelo LPT, normalmente el LPT1
- Un puerto serie (los trazadores gráficos antiguos los suelen usar)
- Un puerto de red (cuando hay conexiones a impresoras de red)
- Un archivo. En este archivo se almacenarán los comandos necesarios para imprimir el documento en la impresora. Se suele usar para generar archivos Postcript.
- Un puerto nuevo, que quizás usa un hardware o software especial. Un ejemplo de este tipo son los puertos Jetdirect de red usados por las impresoras HP.

En este cuadro de diálogo se puede activar la cola de impresión de NT (normalmente esta opción no se activa por defecto, sino que se imprime directamente en el puerto).

Luego hay que elegir el modelo de la Impresora. El cuadro de diálogo está organizado por fabricantes y modelos, por lo que se encuentran rápidamente los modelos adecuados.

También se puede añadir una impresora no disponible en NT con los controladores proporcionados por el fabricante. Tras elegir el modelo hay que ponerle un nombre en el siguiente cuadro de diálogo. Normalmente NT elige un nombre adecuado, que coincide con el modelo de la impresora.

El siguiente cuadro de diálogo permite compartir la impresora. Si se comparte NT se debe elegir:

- Un nombre para la impresora, que también sugiere el asistente.
- Los controladores que se van a compartir. Se puede añadir el soporte para otras plataformas NT y Windows, aunque hará falta tener los discos correspondientes. Esto permite que los clientes se conecten a nuestra impresora sin necesidad de instalar ellos ningún controlador de impresora. Al conectarse a la impresora los controladores se transfieren al cliente de forma automática.

El proceso de instalación finaliza con la impresión de una página de prueba opcional, que permite verificar la correcta instalación de la impresora.

2.5.4 Agregar o quitar programas.

En este icono podemos acceder a un cuadro de diálogo que nos permite realizar varias labores:

2.5.4.1 Instalar nuevas aplicaciones.

El botón instalar lanza un asistente que busca en las unidades de disco y de CD-ROM ficheros con nombre típico de programas de instalación, como pueden ser instalar.exe y setup.exe y si los encuentra nos permite ejecutarlos.

2.5.4.2 Eliminar los programas instalados.

Muchos programas para Windows 95 y NT traen sus propios programas de desinstalación automática. Estos programas saben como registrarse en el registro de NT, de manera que pulsando el botón de agregar o quitar se reinstalan o desinstalan automáticamente.

2.5.4.3 Instalar o eliminar componentes adicionales de NT.

Durante el proceso de instalación, al elegir una de las posibles instalaciones (típica, portátil, a medida...) se seleccionan los componentes que se han de instalar. Con este cuadro de diálogo podemos añadir o eliminar componentes adicionales desde el CD-ROM de instalación.

2.5.5 Desinstalación manual de aplicaciones.

Ocurre a veces que el programa de instalación de una aplicación no es capaz de desinstalarla. Esto puede ocurrir por un fallo del propio programa o porque otra aplicación la ha alterado. En este caso se debe proceder manualmente. Para ello:

- Se deben eliminar todos los ficheros de la aplicación. Normalmente hay que borrar el directorio de la aplicación y algunos directorios de datos.

- Se deben borrar los accesos directos creados por la aplicación. Esto se puede hacer con el explorador de NT, en el escritorio y en el menú de inicio.

- Se deben borrar las librerías de haya dejado la aplicación en el sistema. En NT y 95 al instalar una librería se incrementa la clave del registro:

  HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\SharedDlls

- Se deben eliminar las claves del registro que ha añadido la aplicación, normalmente en:

  HKEY_LOCAL_MACHINE\Software\Compañía\Aplicación y en otras partes del registro.

- La siguiente clave del registro indica el nombre del programa de desinstalación para la aplicación.

  HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Uninstall

2.5.6 Dispositivos.

Este cuadro de diálogo permite ver el estado de los dispositivos instalados en el sistema. NT por defecto instala gran cantidad de dispositivos aunque no los utilice. En particular, el programa de instalación de NT carga los controladores SCSI y algunos controladores de dispositivos de pantalla. Esto es necesario ya que durante el proceso de instalación puede pedir a NT buscar estos dispositivos. Para ello NT carga cada controlador de dispositivos e intenta iniciarlo. Si el controlador de dispositivo arranca quiere decir que el dispositivo está funcionando correctamente.

2.5.6.1 Tipos de inicio para cada dispositivo.

Cada dispositivo puede tener un tipo de inicio:

- *Inicio*. Se utiliza para controladores de dispositivos que se cargan en las primeras etapas de carga del sistema, como por ejemplo el controlador ATAPI y el DISK.

- *Sistema*. Son los dispositivos que se cargan después de cargar el sistema. Por ejemplo el controlador del ratón y de la disquetera (floppy)

- *Automático.* Estos dispositivos se cargan de la carga completa del sistema.

- *Manual.* Estos dispositivos los carga el usuario u otros controladores de dispositivos. Por ejemplo el controlador de la tarjeta de red normalmente lo carga los servicios de red.

- *Deshabilitado.* Son controladores de dispositivos que no se pueden cargar normalmente, como por ejemplo los controladores SCSI de adaptadores no instalados. Hay algunos dispositivos que aparecen como deshabilitados pero están iniciados. Estos dispositivos se cargan en la primera fase de arranque del sistema operativo. Por ejemplo los controladores de sistemas de ficheros FastFAT y NTFS son un ejemplo.

- *Un estado.* El dispositivo puede estar iniciado o no iniciado.

- *Un perfil asociado.* NT permite definir perfiles del sistema diferentes. Cada perfil tiene su propia configuración del sistema. Normalmente se usa para equipos portátiles con unidades de expansión de puertos (docking stations), que permiten añadir periféricos al portátil al insertar el portátil en la ranura de expansión.

2.5.6.2 Dispositivos de cinta.

Este cuadro de diálogo permite agregar dispositivos de cinta. El funcionamiento es análogo al de Adaptadores SCSI, aunque se ha incluido un botón para detectar los dispositivos de cinta instalado.

2.5.6.3 Memoria.

Windows NT y UNIX implementan un direccionamiento lineal de 32 bits y memoria virtual paginada bajo demanda. El VMM se encarga de todo lo relacionado con la política de gestión de la memoria. Determina los conjuntos de trabajo de cada proceso, mantiene un conjunto de páginas libres, elige páginas víctima, sube y baja páginas entre la memoria RAM y el archivo de intercambio en disco, etc.

El cuadro de diálogo de Memoria virtual nos permite ajustar los archivos de paginación y de registro. Este cuadro de diálogo permite ver todos los ficheros de intercambio creados. Para cada unidad de disco se puede establecer un fichero de intercambio, que tendrá un tamaño mínimo y máximo. Este cuadro de diálogo nos indica el total de espacio de intercambio disponible para el sistema y las aplicaciones.

También en este cuadro de diálogo se puede establecer el tamaño del registro máximo. Esto evita que el mal funcionamiento de una aplicación pueda desbordar el registro. Normalmente no hace falta modificar el tamaño sugerido por el sistema.

El Windows NT utiliza un proceso llamado demanda de página para intercambiar datos entre la RAM y uno o más páginas memoria. Cuando el Windows NT es instalado crea una memoria virtual de estas páginas de memoria llamada PageFile.SYS.

Para acceder a este ítem se hace click dentro del panel de control propiedades del sistema y luego en la barra memoria virtual.

2.5.7 Administrador de procesos.

Se encarga (en colaboración con el administrador de objetos) de crear, destruir y gestionar los procesos y subprocesos. Una de sus funciones es la de repartir el tiempo de CPU entre los distintos subprocesos. Suministra sólo las relaciones más básicas entre procesos y subprocesos, dejando el resto de las interrelaciones entre ellos a cada subsistema protegido concreto. Por ejemplo, en el entorno POSIX existe una relación filial entre los procesos que no existe en Win32, de manera que se constituye una jerarquía de procesos. Como esto sólo es específico de ese subsistema, el administrador de objetos no se mete en ese trabajo y lo deja en manos del subsistema [Wayatt, 1998].

2.5.8 El administrador de entrada/salida (I/O Manager).

Consta de varios subcomponentes: el administrador del sistema de ficheros, el servidor de red, el redirector de red, los drivers de dispositivo del sistema y el administrador de cachés. Buena parte de su trabajo es la gestión de la comunicación entre los distintos drivers de dispositivo, para lo cual implementa una interfaz bien definida que permite el tratamiento de todos los drivers de una manera homogénea, sin preocuparse del funcionamiento específico de cada uno. Trabaja en conjunción con otros componentes del Executive, sobre todo con el VMM. Le proporciona la E/S síncrona y asíncrona, la E/S a archivos asignados en memoria y las caches de los ficheros. El administrador de caches no se limita a gestionar unos cuantos buffers de tamaño fijo para cada fichero abierto, sino que es capaz de estudiar las estadísticas sobre la carga del sistema y variar dinámicamente esos tamaños de acuerdo con la carga. El VMM realiza algo parecido en su trabajo.

2.5.9 Administración de discos.

Para llegar al administrador de discos, debemos hacer click en inicio, luego señalamos programas y luego el menú herramientas administrativas (común). Allí encontraremos el administrador de discos el cual nos permite administrar los recursos del disco, realizar cambios al disco duro, o para crear particiones en un disco duro adicional.

2.5.10 Command prompt.

Windows NT permite al administrador y el cliente manejar las aplicaciones bajo este sistema operativo utilizando otras herramientas como el command prompt.

Command Prompt. Se encuentra en inicio y programas. Desde el Command Prompt un usuario puede hacer entre otras cosas lo siguiente:

- Ejecutar cualquier archivo batch, ya sea ".bat", o bien, ".cmd"
- Ejecutar cualquier Windows NT (32-bit), Windows 3.x (16-bit), Ms-Dos, OS/2 1.x.

- Administrar o usar los recursos de la red.
- Copia y pegar información entre las aplicaciones, aún cuando las aplicaciones corran bajo distintos subsistemas.

Windows NT también nos permite configurar el Command Prompt, mediante las propiedades de la consola de la ventana, de esta manera da prioridad a las aplicaciones y distribuye el tiempo de proceso entre las mismas. La base de la prioridad puede ser ajustada para aumentar o disminuir el desempeño.

Los niveles de las prioridades van desde 0 a 31.

Prioridad

0-15 aplicaciones dinámicas

16-31 aplicaciones de tiempo real.

2.5.10.1 Comenzar una aplicación con una prioridad específica.

Para cambiar la base de la prioridad de una aplicación se utiliza el comando Start y alguna de estas opciones.

/realtime (establecer la base de la prioridad a 24)

/high (establecer la base de la prioridad a 13)

/normal (establecer la base de la prioridad a 7)

/low (establecer la base de la prioridad a 4).

EJEMPLO
Si se quiere comenzar el Notepad con una prioridad alta, en el Command Prompt

Star/High Notepad

2.5.11 Copias de resguardo.

Para realizar copias de seguridad, el proceso es muy sencillo, solo hacer click en el botón inicio, señalar a programas y luego a herramientas administrativas (común). Finalmente hacer click en copias de seguridad.

Desde esa ventana se puede seleccionar los archivos de los que desea hacer copias de seguridad, comprobar el estado de la copia de seguridad y establecer diversas opciones.

2.5.12 El administrador de usuarios.

En NT hay dos tipos de usuarios, aquellos que pertenecen a una máquina que corre NT Workstation o Server y aquellos que pertenecen a un dominio NT. Para cada uno de estos tipos de usuarios existe una herramienta de administración: el administrador de usuarios incluido en NT Workstation y el administrador de usuarios para dominios incluido en NT Server. El funcionamiento de ambos es muy similar, casi idéntico pero el administrador de usuarios para dominios dispone de más opciones. Por ello, se describirá el administrador de usuarios para dominios.

Las tareas que se pueden realizar con el administrador de usuarios son las siguientes:

- Añadir, modificar y eliminar usuarios del dominio.
- Añadir, modificar y eliminar grupos locales y globales del dominio.
- Fijar el plan de cuentas y contraseñas en el dominio.
- Fijar la política de derechos de usuario en el dominio.
- Establecer el sistema de auditoria en el dominio.
- Establecer relaciones de confianza entre dominios.

2.5.12.1 Creación y modificación de usuarios en el dominio.

Para crear un usuario se pulsa Usuarios\Nuevo y se completa el cuadro de diálogo. *Usuario nuevo,* en este cuadro hay que rellenar una serie de campos:

- Nombre de usuario: Es el identificador que representa al usuario en el dominio. Debe ser una palabra completa, sin espacios en blanco ni caracteres especiales. Este identificador debe ser único en el dominio. Se le suele conocer también como nombre de cuenta o login. Este identificador es el que se debe suministrar, junto con la contraseña, para iniciar sesión en un dominio NT.

- Nombre completo: Es el nombre completo del usuario. Si este usuario es una persona se suele escribir el nombre y apellidos.

- Descripción. Conviene añadir una descripción de la labor, grupo de personas o departamento al que pertenecen el usuario, sobre todo si la base de datos de usuarios en el dominio es grande.

- Contraseña. Aquí se debe escribir la contraseña para el usuario. Puede incluir espacios, mayúsculas y minúsculas, e incluso caracteres especiales. NT admite hasta 14 caracteres. Cuando se va escribiendo aparecen asteriscos, y una vez completada en el cuadro aparece una línea de asteriscos, siempre de la misma longitud.

- Repetir contraseña. Hay que introducir de nuevo la contraseña, para comprobar que se ha escrito correctamente.

Tras estos campos aparecen una serie de botones activables:

- ***El usuario debe cambiar su contraseña***. La primera vez que inicia sesión en NT se va a solicitar que introduzca una nueva contraseña. Habitualmente los administradores crean los usuarios nuevos con contraseñas del tipo especial que obligan al nuevo usuario a cambiar su contraseña al iniciar sesión por primera vez.
- ***El usuario no puede cambiar su contraseña***. Se utiliza en tareas administrativas especiales, y bloquea el cambio de contraseña por parte del usuario.
- ***La contraseña nunca caduca***. Desactiva la caducidad de contraseñas para esta usuario. Se utiliza normalmente sólo para algunos usuarios que lo necesitan.
- ***Cuenta desactivada***. Permite bloquear la cuenta fácilmente sin borrarla. Se suele utilizar cuando el usuario no va a iniciar sesión durante un periodo de tiempo o cuando se va a cambiar la configuración o entorno del usuario y se necesita que no pueda acceder temporalmente al sistema.

2.5.12.2 Administrador de usuarios, usuario nuevo, grupos.

Al final del cuadro de diálogo aparecen varios botones:

- Grupos. Permite establecer los grupos a los que pertenece un usuario. En NT hay dos tipos de grupos:
  1. *Grupos globales*. Válidos para dominios en los que se confian. Aparecen marcados con el icono de grupo global.
  2. *Grupos locales*. Son grupos locales al servidor o estación de trabajo.
- Además se puede asignar un grupo primario para el usuario [Este grupo es utilizado en entornos Macintosh]

2.5.12.3 Administrador de usuarios, usuario nuevo, perfiles.

- *Perfil*. Permite controlar las características del entorno de un usuario.

  Se puede establecer nombre del fichero que representa el perfil del usuario para NT. Hay que escribir un fichero en formato UNC (Univer name convention), es decir:

  ```
  \\servidor\recurso\directorio\fichero.bat
  ```

- *Archivo de inicio*. Asigna un archivo que se ejecutará al iniciar la sesión de red en el dominio. El archivo debe residir en el servidor que valida el inicio de sesión. Este fichero se debe hallar en la carpeta NETLOGON del servidor que valida el inicio de sesión.

- *Directorio de trabajo*. Admite dos modalidades de uso.

- *Ruta de acceso local*. Utilizar una ruta local al ordenador en que se inicia la sesión.

- *El directorio de trabajo*. Es el que aparece por defecto en los cuadros de diálogo de guardar un fichero en una aplicación Windows.

- *Horas*. En el botón Horas podemos acceder al cuadro de diálogo de Horas de inicio de sesión.

2.5.12.4 Administrador de usuarios, usuario nuevo, horas de inicio.

En este cuadro de diálogo se pueden fijar las horas de inicio de sesión en los servidores del dominio, en intervalos de 1 hora, para cada día de la semana. En principio una vez iniciada la sesión el usuario puede seguir trabajando en los servidores, aunque se puede modificar esto para que los servidores cierren la sesión del usuario al terminar las horas permitidas, mediante el cuadro de diálogo Plan de cuentas en el menú de Directivas del Administrador de Usuarios.

Iniciar desde. Este cuadro de diálogo permite seleccionar los ordenadores desde los cuales un usuario puede iniciar sesión. Se pueden especificar hasta 8 ordenadores que ejecuten NT, o permitir el inicio en todos los ordenadores del dominio.

2.5.12.6 Administrador de usuarios, usuario nuevo, cuenta.

El cuadro de diálogo Información de cuenta permite especificar el tipo de cuenta y su duración. Se puede elegir que la cuenta caduque en una fecha determinada o que no tenga caducidad. También se puede especificar si es una cuenta global o local.

2.5.12.7 Administrador de usuarios, usuario nuevo, marcado.

El cuadro de diálogo de Marcado permite especificar las propiedades de marcado para el usuario.

Modificar un usuario.

Utilizando el menú Usuario\Propiedades o haciendo doble clic sobre un usuario o grupo se puede modificar el mismo. Al modificar un usuario se acceden a los mismos cuadros de diálogo empleados al crearlo salvo que ahora aparece una casilla para cuentas bloqueadas. Si el bloqueo de cuentas está activado en el dominio y el usuario ha fallado el número de veces limitado para ese dominio, esta casilla aparece activada. Al desactivarla, la cuenta del usuario es desbloqueada.

Nota importante: existe una ligera demora al cambiar las propiedades de un usuario o grupo del dominio, debido a que la base de datos de usuarios del dominio tarda unos minutos en actualizarse en los controladores secundarios. Se puede forzar la actualización inmediata mediante la sincronización manual de los servidores.

2.5.13 Creación de grupos.

Para un Windows NT el administrador de usuarios permite crear grupos locales, que sólo tienen validez en el propio ordenador. El administrador de usuarios para dominios permite crear dos tipos de grupos: globales y locales.

Los grupos locales pueden obtener permisos en los servidores del dominio propio. Pueden ser miembros de los grupos locales los miembros del dominio, los grupos globales del dominio y los grupos globales de otros dominios en los que se confía. Los grupos globales tienen como miembros a los usuarios del dominio y se pueden utilizar tanto en los servidores del dominio como en las estaciones de trabajo del dominio. También se pueden usar en otros dominios en los que se confía.

2.5.13.1 Creación de un grupo global nuevo.

Para añadir un grupo global nuevo se selecciona el menú Usuario\Grupo Global nuevo y se proporcionan en el cuadro de diálogo el nombre del grupo y una descripción opcional. Podemos además añadir usuarios al grupo.

2.5.13.2 Creación de un grupo local nuevo.

Para añadir un grupo local nuevo se selecciona el menú Usuario\Grupo Local nuevo y se proporcionan en el cuadro de diálogo el nombre del grupo y una descripción opcional. Podemos además añadir usuarios al grupo.

2.5.13.3 Usos de grupos locales y globales.

Cuando se crea un dominio el programa de instalación de NT crea una serie de grupos globales y locales del dominio. También se crea la cuenta del administrador del dominio, y se añade al grupo administradores del dominio.

El grupo administradores que se ha creado permite administrar todos los servidores del dominio. NT añade al grupo local administradores el grupo administradores del dominio. De igual manera ocurre con el grupo de usuarios e invitados.

Cuando una estación de trabajo es añadida al dominio, NT añade automáticamente los tres grupos globales del dominio (administradores, usuario e invitados del dominio) a los grupos locales correspondientes de la estación de trabajo.

Los grupos locales se utilizan para asignar tareas especiales en las estaciones de trabajo o servidores del dominio. Por ejemplo se pueden crear los grupos especiales para trabajar con el recurso como una impresora láser en color, cuyo acceso queremos restringir. Uno lo podemos llamar "Administradores de láser color" y otro "Usuarios de láser color". Ahora basta dar permisos de impresión al grupo "Usuarios de láser color" y control total al grupo "Administradores de láser color".

Para añadir usuarios a estos grupos bastaría añadir al grupo "Administradores de láser color" el grupo "Administradores del dominio" y al grupo "Usuarios de láser color" los miembros del dominio autorizados a imprimir en ella. Este último paso lo podemos hacer de un medio más eficiente si creamos un grupo global "Usuarios de impresora láser color" y añadimos este grupo al grupo local "Usuarios de láser color". Para dar permisos de

impresión a los usuarios y grupos utilizaremos el Administrador de Impresión de dicha impresora, accesible desde el menú de inicio Configuración\Impresoras.

Cada usuario en NT debe pertenecer a uno o varios grupos globales o locales, indistintamente. Los grupos locales se definen en cada estación de trabajo NT o en cada servidor. Para los servidores hay dos posibles configuraciones: si el servidor está configurado como controlador de dominio, primario o secundario, los grupos locales son los que se definen para todos los servidores controladores del dominio. Es decir, todos los controladores del dominio comparten la misma lista de grupos locales. Para los servidores configurados como servidor, los grupos locales se definen en el propio servidor, del mismo modo en que se hacen con las estaciones de trabajo.

Finalmente como pudimos observar, Windows NT proporciona una manera sencilla de administración, puesto que la mayoría de las funciones se realizan desde el menú y dentro de un ambiente gráfico, el cual proporciona una facilidad extraordinaria para realizar las tareas administrativas.

2.6 TRABAJO EN RED

Una de las ventajas más importantes que ha tenido siempre NT es la elasticidad que tiene a la hora de integrarse con otros equipos en red, de hecho convive sin problemas con redes basadas en sistemas Novell, AppleTalk, UNIX, SNA y cualquier cosa que se le integre, sirviendo incluso de pasarela entre estos mundos tan distintos. Este comportamiento tan versátil se debe a la gran cantidad de protocolos que incorpora y que pueden convivir simultáneamente en un sistema con Windows NT. Para cualquier administrador de sistemas es obvio que si necesita conexión con Novell habrá de instalar el protocolo IPX/SPX, para convivir con UNIX echará mano de TCP/IP y si los Macintosh abundan tendrá que usar AppleTalk. Lo que a veces no queda tan claro es el protocolo a utilizar en una red basada únicamente en sistemas operativos de Microsoft (MSDOS, Windows 3.x, Windows 95 y el propio Windows NT), es aquí donde pretendemos poner un poco de luz.

Protocolos "de serie": IPX/SPX y NetBEUI. Al instalar Windows NT los protocolos que se instalan por defecto son IPX/SPX y NetBEUI. Aunque la elección de IPX/SPX me parece acertada no entiendo porqué se nos endosa NetBEUI cuando la propia Microsoft nos dice que no le parece un protocolo adecuado para una red que se dice serlo. Vamos a empezar a analizar este primer protocolo:

2.6.1 NetBEUI.

Es el protocolo utilizado por las antiguas redes basadas en Microsoft LAN Manager. Es muy rápido en pequeñas redes que no lleguen a la decena de equipos y que no mueven ficheros de gran tamaño, a partir de ahí es mejor otra opción y se debe desinstalar de los clientes y servidores, esto último siempre que no se tenga ningún equipo que utilice LAN Manager.

2.6.2 IPX/SPX.

Este protocolo, implementado por Novell, ha demostrado sobradamente su flexibilidad en redes de área local, es rápido, fácil de configurar y requiere pocas atenciones. Es el protocolo que Microsoft recomienda para redes de área local basadas en DOS, Windows 3.x, Windows 95 y Windows NT. El principal inconveniente que presenta para redes medianas y grandes es que no se puede "enrutar" o sea que no puede pasar de una subred a otra si entre ambas hay un ruteador, por lo que no puede usarse en redes WAN. Otro inconveniente que presenta en redes con un cierto número de equipos es que puede llegar a saturar la red con los broadcast que lanzan los equipos para anunciarse en la red.

2.6.3 TCP/IP.

Este protocolo juega aquí con ventaja pues se trata de uno de los favoritos y además se hace imprescindible si se está conectado a Internet o se quiere crear una intranet. La capacidad de TCP/IP para mover información en una red, por grande que sea, sin perder datos, su sistema de nombres y direcciones, y su facilidad para saltar de una red a otra lo convierten en el candidato ideal para cualquier red de ordenadores dispuesta a no quedarse dentro de las paredes de un edificio. No obstante pueden achacársele algunos inconvenientes como la dificultad de configuración para el usuario y la necesidad de un mantenimiento constante por parte del administrador de la red.

El primer inconveniente se debe a la necesidad que tiene el usuario de conocer algunos datos imprescindibles antes de que el sistema empiece a funcionar en red: dirección IP, máscara de red, dirección del servidor de nombres y dirección del ruteador, afortunadamente este problema puede resolverse utilizando el servicio de configuración dinámica de equipos (DHCP), que viene incluido en Windows NT Server, este servicio asigna los datos mencionados arriba a cada equipo en el momento en que este se conecta en red de manera transparente para el usuario. El trabajo de mantenimiento por parte del administrador tampoco es insignificante: asignación de direcciones IP a los nuevos equipos, mantenimiento de la tabla de nombres en el servidor de nombres si este existe o, peor aún, en cada equipo si no existe y vigilar que no haya direcciones duplicadas por citar sólo algunos. De nuevo NT Server nos hecha una mano si combinamos la potencia de DHCP con el servicio de nombres para Windows (WINS) y el reciente servicio de nombres de dominio (DNS).

Otro inconveniente que aún no hemos mencionado es la falta de seguridad de TCP/IP frente a los "hackers" que tengan acceso físico a la red, ya que las tramas TCP/IP no van codificadas y con un software adecuado podría capturarse parte de la información que estamos enviando. Para este problema comienzan a surgir soluciones como el protocolo punto a punto (PPTP), que encripta las tramas TCP/IP que enviamos, estableciendo de esta forma un canal seguro incluso a través de Internet.

2.6.4 NetBIOS.

NetBIOS un escalón más arriba. No podemos nunca hablar de NetBIOS como una alternativa a los protocolos mencionados arriba, pues se trata de un protocolo que se

encuentra un escalón más arriba que los anteriores (más cerca del usuario). NetBIOS es un intermediario entre dichos protocolos y nuestras aplicaciones, que nos permite conectarnos al resto de los equipos de nuestra red usando nombres sencillos y fáciles de recordar (SERVIDOR, COMPRAS, ANDROMEDA) sin importar el protocolo de red que estemos utilizando para comunicarnos con ellos. De esta manera para el usuario la red se convierte en algo transparente por la que puede navegar usando el icono "Entorno de red".

Además no tenemos que preocuparnos nunca de él pues se instalará automáticamente sobre los protocolos que configuremos en nuestro equipo.

2.6.5 Conclusión sobre el protocolo de red.

La recomendación es que se mantenga la red funcionando con el menor número de protocolos posibles, siendo muy conveniente que haya homogeneidad entre los equipos que vayan a compartir recursos entre si, ya que las máquinas que no tengan al menos un protocolo común no podrán verse entre si ni compartir recursos.

Si por motivos de tamaño no se utiliza el NetBEUI, la elección queda entre IPX/SPX y TCP/IP, utilizar el primero en una red local sin acceso a Internet y el segundo en una red no local (WAN,MAN) o si se necesita Internet o intranet. Si, como es habitual, en la red hay máquinas que no salen a Internet y otras que si lo hacen, se puede instalar en todas IPX/SPX y en las que acceden a Internet se añade TCP/IP, de esta manera las máquinas que no acceden a Internet están más seguras y todas se ven entre si.

2.7 LA SEGURIDAD DEL SISTEMA WINDOWS NT

La seguridad en Windows NT es una combinación de técnicas que aseguran un nivel de protección consistente contra los accesos no deseados. Para implementar la seguridad, tendremos que proteger la red, el sistema operativo y los datos. Para eso, disponemos de la autentificación de acceso propia de Windows NT, seguridad a nivel de objeto y derechos de usuarios. Para sacar provecho de los más altos niveles de seguridad que permite Windows NT, el nivel de seguridad C2, necesitaremos tanto el hardware como el software adecuados. NT dispone de herramientas de auditoría que nos permitirán conocer nuestros niveles de seguridad, pero tendremos que tener muy presentes los temas relativos a la seguridad cuando entran en juego las comunicaciones sobre la Internet. Para estar seguro que estamos protegidos en todos los frentes, es necesario conocer determinadas técnicas.

2.7.1 Clasificación de la seguridad.

La seguridad puede ser clasificada en tres diferentes áreas funcionales: seguridad a nivel de red, seguridad del sistema operativo y encriptación de datos.

2.7.1.1 La seguridad de red.

La Seguridad de red ofrece autentificación (verificando que el servidor de datos y que el receptor de los mismos son correctos) y verificando la integridad de la información (de forma que los datos enviados y los recibidos sean los mismos). Conseguir este nivel de

seguridad a nivel de red significa haber implementado un protocolo de red, como NetBEUI o TCP/IP, ajustado a las necesidades de la red.

Esos protocolos ofrecen varios niveles de seguridad, rendimiento (conseguidos reduciendo al mínimo la carga derivada de la seguridad), flexibilidad, y disponibilidad sobre múltiples plataformas.

Tras haber definido e instalado una determinada infraestructura de red, añadir y extender protocolos de seguridad es algo teóricamente muy simple. Todo lo que se necesita es llegar a un consenso entre los miembros del equipo.

2.7.1.2 Seguridad a nivel de sistema.

La seguridad a nivel de sistema operativo debe estar integrada con el mismo desde el principio. Si esas funciones básicas de seguridad no han sido implementadas al propio sistema operativo desde un principio, implementarlas con posterioridad será casi imposible. Por ejemplo, Microsoft no fue capaz de implementar una seguridad seria a sus versiones de 16 bits de Windows tras su fase de desarrollo. Fue necesario un nuevo sistema operativo de 32 bits, y un nuevo modelo de programación (la API Win32) para poder hacerlo. Windows NT dispone de unas robustas funciones de seguridad que controla el acceso de los usuarios a los objetos como archivos, directorios, registro de sistema e impresoras.

También incluye un sistema de auditoría que permite a los administradores rastrear los accesos a los archivos u a otros objetos, reintentos de inicio de sesión, apagados y encendidos del sistema, etc... En cambio, Windows 95 dispone únicamente de un rudimentario sistema de seguridad en el inicio de sesión, y no dispone de seguridad a nivel de objetos.

2.7.1.3 Encriptación de datos.

Encriptación de datos. Puede operar de distintas formas. Muchas aplicaciones disponen de encriptación por sí mismas. Algunos protocolos, como SSMTP (Secure Simple Mail Transfer Protocol) soportan encriptación automática. La encriptación ofrecida por terceras compañías, como PGP (Pretty Good Privacy) también está disponible. Incluso Microsoft ha añadido un sistema de encriptación básico, CAPI (Cryptography API) a la API Win32 [Wayatt, 1998].

CAPI consiste en un juego de funciones que permiten a las aplicaciones y a los desarrolladores de sistemas de software acceder de forma independiente a los servicios de criptografía. NT dispone de un servicio básico de criptografía que nos permite codificar los datos facilitando así el almacenamiento seguro y una transmisión segura combinando claves públicas y privadas. El método de encriptación es similar al PGP.

2.7.2 Aspectos de la seguridad NT.

NT ofrece seguridad en tres áreas fundamentales. Se trata de autentificación en el inicio de sesión, seguridad a nivel de objetos y derechos de los usuarios.

La "Local Security Authority" efectúa validaciones interactivas y remotas (a través del RAS), inicios de sesión locales y globales (en dominios) verificándolo contra SAM (Security Account Manager), la base de datos donde se almacenan los nombres de los usuarios y sus contraseñas. La "Local Security Authority" también gestiona los mensajes de auditoría.

El "Security Reference Monitor" verifica si un usuario tiene derecho a acceder a un objeto y ejecutar la acción solicitada. Además, es el responsable de los mensajes de auditoría. Con el diálogo permisos del Administrador de archivos (si estamos en NT 3.51) o el Explorador de archivos (en Windows NT 4.0), podemos controlar la seguridad de la mayoría de los objetos. Por ejemplo, la activación y el acceso al objeto servidor del DCOM (en Windows NT 4.0) están completamente integrados con el modelo de seguridad de Windows NT.

Aparte de la seguridad de los objetos, el NT permite controlar, y monitorizar funciones de sistema. Con el Administrador de usuarios podemos controlar cuales cuentas de usuario o grupo pueden, por ejemplo, añadir estaciones de trabajo a un dominio, salvar o restaurar archivos y directorios, cambiar la hora del sistema, iniciar una sesión localmente, gestionar los registros de auditoría y seguridad y cerrar el sistema. La Local Security Authority mantiene la Planificación de cuentas de usuario.

2.7.3 Administración de cuentas: Autentificación de inicio de sesión.

Windows NT gestiona tanto a usuarios como máquinas. De forma que debemos validar a los usuarios autorizados y proveerlos de lo necesario para acceder al sistema.

A un nivel general, un dominio NT es una colección de máquinas, a las cuales el controlador de dominio administra como si se tratase de una única máquina, compartiendo una misma base de datos de seguridad. Dicha base de datos mantiene información de todos los usuarios y grupos de ese dominio. Una cuenta de dominio, llamado tambien cuenta global, tiene el formato dominio\usuario. Si iniciamos una sesión en una máquina del dominio e intentamos conectarnos a una unidad de red, tendremos que introducir nuestros datos con ese formato.

Además de las cuentas de usuarios, cada máquina puede disponer de una base de datos de seguridad local que contendrá la información de los usuarios y grupos de usuarios exclusivos de ese sistema. De hecho, las máquinas locales funcionan como dominios independientes. Las cuentas de usuarios o grupos en esas máquinas locales tiene el formato maquina\usuario y serán exclusivas de esa máquina.

La herramienta básica para administrar los usuarios y grupos de un dominio es el programa USRMGR.EXE, conocido como *Administrador de Usuarios para Dominios*. Windows NT Workstation incluye una versión reducida de este programa y permite la gestión únicamente de esa máquina. Con este programa podemos visualizar información local y del dominio, dependiendo de si añadimos parámetros o no al ejecutarlo.

Ejecutar USRMGR sin parámetros, gestionará el dominio al cual el usuario esté conectado en ese momento. Si ejecutamos *USRMGR nombre_dominio* como parámetro, se gestionará ese dominio en concreto, y si se ejecuta *USRMGR \\nombre_máquina*, se gestionarán los usuarios de esa máquina en concreto.

Cargar los dominios puede ser cuestión de minutos si se trata de organizaciones con muchos dominios, por lo que se agradece a la larga la posibilidad de parametrizar y gestionar un dominio o máquina en concreto. El programa también permite a los administradores gestionar las relaciones de confianza de los dominios, de forma que los usuarios del dominio A puedan acceder a los recursos del dominio B como si fueran usuarios del segundo.

Un valor único y permanente, el SID (Security Identifier) identifica usuarios individuales y grupos de usuarios. Después de que el sistema asigne un SID, no lo volverá a utilizar, por lo que si por lo que sea borramos un usuario o grupo y más tarde nos vemos en la necesidad de volverlo a crear, se le asignará un nuevo SID. Será necesario volver a reestablecer de nuevo los derechos de acceso sobre los objetos para ese nuevo SID.

2.7.4 El nivel de seguridad C2.

De acuerdo con la TCSEC (Trusted Computer System Evaluation Criteria) publicado por la NCSC (National Computer Security Center) [Ezzell, 1997], la seguridad C2 requiere una específica combinación y configuración de software y hardware. NT no es en sí, un sistema operativo que cumpla con el nivel C2 de seguridad, pero podemos hacer que lo sea. Microsoft diseñó esta posibilidad dentro del modelo de seguridad del NT desde los primeros momentos. El nivel C2 de seguridad requiere lo siguiente:

- Cada recurso tiene un propietario que debe controlar el acceso al recurso.
- Cada usuario debe iniciar la sesión con un identificador y contraseña exclusivos antes de poder acceder al sistema.
- El sistema monitoriza toda actividad de los usuarios.
- Los administradores de sistema deben auditar los eventos relativos a la seguridad del sistema, y el acceso a los datos de auditoría deben ser seguros.
- El sistema debe poder protegerse a sí mismo de interferencias externas.
- El Resource Kit de Windows NT incluye el programa C2CONFIG.EXE (C2 Configuration Manager), que verifica que el sistema cumpla con los requisitos C2.

Capitulo 4

El Sistema Operativo UNIX

3.1 INTRODUCCIÓN

En este capítulo hablaremos de UNIX, uno de los sistemas operativos más populares en el mundo. Robusto, seguro y confiable, UNIX siempre se ha considerado entre uno de los sistemas operativos más estables y es por ello que muchas empresas e instituciones en el mundo lo prefieren como plataforma de trabajo. Un gran porcentaje de los servidores que actualmente constituyen la red Internet se encuentran montados sobre este sistema. Es aquí donde se construyeron los protocolos que permitieron la conectividad abierta de los sistemas para dar lugar a lo que ahora llamamos Internet. UNIX ha sido y seguirá siendo líder de sistemas operativos, aunque a decir verdad, para muchos UNIX representa un sistema viejo y poco amigable, pues UNIX se ha mantenido casi sin cambios en su estructura básica durante muchos años, a pesar de esto, UNIX se ha distinguido por presentar un diseño excelente en cuanto a su esquema de trabajo.

3.1.1 Introducción a UNIX

UNIX es un sistema operativo multiusuarios. En un sistema UNIX cada quien tiene su propia identidad y puede crear sus propios archivos en forma privada; pero también puede ver lo que los demás (usuarios) están haciendo. Se puede compartir archivos con ellos, mandarles mensajes e interactuar con ellos. UNIX es un sistema multitareas. Se pueden correr varios programas al mismo tiempo, y mandar mensajes entre ellos.

El diseño original de UNIX fue un sistema de un solo usuario, pero permitía múltiples tareas o procesos para ser cargados en memoria y compartir el tiempo del procesador central. Una vez que sus creadores lograron esto, fue relativamente fácil extender los conceptos para que las diferentes tareas pudieran ser asignadas a diferentes usuarios y a diferentes terminales.

UNIX es un sistema operativo de propósitos generales. Actualmente ha logrado un gran éxito en dos áreas diferentes: en el sistema multiusuario, para un departamento o una compañía pequeña soportando cuentas, automatización de oficina y bases de datos. En el desempeño de un usuario de workstation para gráficas, científica o ingeniería de computación

UNIX al igual que otros sistemas operativos, fue creado para satisfacer las necesidades de los usuarios que demandaban un mejor sistema operativo capaz de realizar las tareas más difíciles de una manera más fácil y con menos recursos. UNIX desde sus inicios supero estas expectativas y demostró ser capaz de realizar tareas más rápidamente, soportar grandes cargas de trabajo y facilitar la conectividad entre computadoras que hasta entonces había sido un objetivo difícil de lograr para otros sistemas operativos.

A pesar de que UNIX ha cobrado gran fuerza y ha sido categorizado como uno de los sistemas operativos más robustos y completos hasta nuestros días, es importante hacer referencia sobre lo que es realmente este sistema operativo:

UNIX en un sentido estricto es el núcleo de un sistema operativo de tiempo compartido, un programa que controla los recursos de una computadora y los asigna entre los usuarios. Permite a los usuarios correr sus programas, controla los dispositivos periféricos conectados a la máquina y proporciona un sistema de archivos que administra el almacenamiento a largo plazo de información, tal como programas datos y documentos. En un sentido más amplio, UNIX no solo es el núcleo sino también programas esenciales; entre ellos, compiladores, editores, lenguajes de comandos, programas para copiado e impresión de archivos.

Así mismo, UNIX proporciona a los usuarios diferentes herramientas y utilidades que se pueden utilizar para realizar una gran variedad de trabajos. Algunas de estas herramientas son órdenes simples que se pueden utilizar para llevar a cabo tareas específicas. Otras herramientas y utilidades son realmente pequeños lenguajes de programación que se pueden utilizar para construir guiones que resuelvan problemas. Lo más importante es que las herramientas están diseñadas para funcionar juntas, como partes de una máquina o bloques de construcción.

Por otro lado, UNIX cuenta con capacidades multiusuario, puede ser utilizado por computadoras con muchos usuarios o con un único usuario. También es un sistema operativo multitarea, ya que un único usuario puede llevar a cabo más de una tarea al mismo tiempo. Además de esto, se proporciona un entorno excelente para redes, UNIX ofrece programas y facilidades que proporcionan los servicios necesarios para construir aplicaciones basadas en red, base de la computación distribuida. El sistema operativo UNIX ha demostrado ser útil en los ambientes de trabajo cliente/servidor en los cuales las máquinas de red pueden ser al mismo tiempo clientes y servidores. Este sistema ha sido pionero en el desarrollo de los servicios de Internet y ha colaborado en gran parte a su crecimiento.

3.1.1.1 Componentes principales de un sistema operativo UNIX

Para tratar de comprender la forma en que trabaja un sistema UNIX, es necesario entender su estructura. Básicamente está compuesto por los siguientes componentes: el núcleo, el shell, el sistema de archivos y las órdenes o programas.

Las relaciones que existen entre los diferentes componentes de UNIX y el hardware se observan en la figura 3-1.

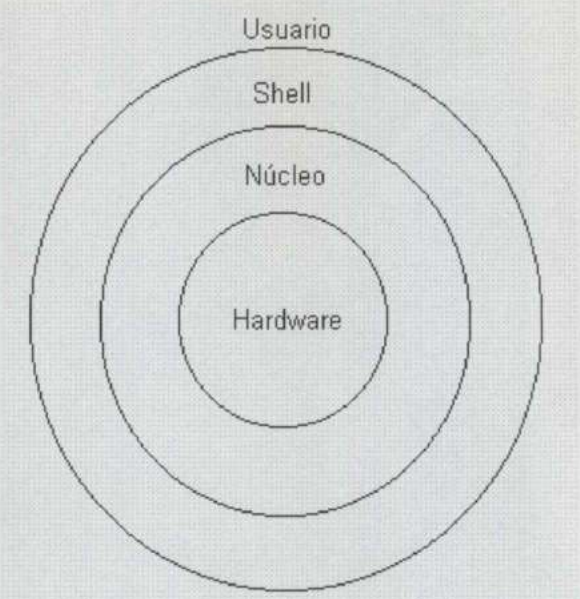

Figura 3-1 Relaciones entre componentes de UNIX

3.1.1.2 Programas y utilidades UNIX

En este sistema operativo se pueden utilizar aplicaciones construidas basadas en órdenes, herramientas y programas del sistema. Los programas de aplicación llevan a cabo diferentes tipos de tareas, algunos de ellos llevan a cabo funciones generales que pueden ser utilizadas por una gran variedad de usuarios, pues son creadas con este propósito. Las aplicaciones se pueden dividir en dos grandes categorías llamadas: aplicaciones horizontales y aplicaciones verticales [Rosen, 1997], la primera de ellas se refiere a los programas tales como procesadores de texto, hojas de cálculo, sistemas de gestión de bases de datos, compiladores, etc., los cuales son utilizados por usuarios de las ramas de la industria, la educación y el gobierno. Por otro lado existe la segunda categoría que incluye exclusivamente software de la industria como paquetes de administración, por ejemplo de bancos u hoteles.

En los últimos años cierto tipo de aplicaciones ha experimentado un crecimiento mayor, una de las más importantes son las aplicaciones de red, debido al gran crecimiento que está teniendo la red mundial Internet, por lo que lo usuarios demandan cada vez mayores y mejores servicios que cubran sus necesidades. Así mismo, las aplicaciones multimedia son otra categoría que actualmente esta teniendo un enorme crecimiento.

UNIX esta integrado por cientos de utilidades y programas que ayudan al usuario a realizar sus tareas, estos programas se pueden utilizar de manera independiente realizando la función para la cual fueron diseñados; así también pueden ser utilizados en conjunto para llevar a cabo tareas más complejas. En UNIX como en otros sistemas operativos existen miles de aplicaciones, muchas de estas comerciales, las cuales se pueden adquirir y están

disponibles a través de proveedores de software, sin embargo las hay también en un gran número en los sitios de software libre en la red Internet.

3.1.1.3 El sistema de archivos, el shell y el núcleo de UNIX

Al igual que otros sistemas operativos, el sistema UNIX cuenta con una organización jerárquica a través de un sistema de archivos, es una forma de muy simple pero también muy eficiente. En primer lugar debemos mencionar que la unidad básica de organización es llamada archivo, este concepto básico toma un significado muy diferente en otros entornos de trabajo, pero por el momento no vamos a profundizar mucho en él, solo cabe mencionar que para este sistema operativo tanto los archivos de datos, los directorios, los dispositivos tales como impresoras, terminales y dispositivos de almacenamiento entre otros, son vistos con el mismo enfoque y son manipulados con las mismas órdenes dentro del shell.

El sistema de archivos en UNIX proporciona una manera fácil para organizar, almacenar, recuperar, manipular y gestionar la información; de tal manera que dichos archivos se encuentran organizados en un árbol jerárquico agrupados por directorios; esto ofrece una ventaja considerable, pues los archivos de datos y los archivos de dispositivos se encuentran formando parte del mismo árbol de directorios. Este árbol puede crecer según sea necesario y se pueden agregar dispositivos del sistema en cualquiera de sus ramas así como archivos de datos y directorios.

El Shell en UNIX es un programa que hace la función de interprete de comandos, algo similar a los que maneja MS-DOS con al archivo "command.com" del sistema operativo. El Shell lee las órdenes y las interpreta como peticiones de ejecución de un programa, lo cual realiza posteriormente. Debido a esto se le denomina interprete de comandos u órdenes, sin embargo a diferencia de otros sistemas operativos, en UNIX el shell también se utiliza como lenguaje de programación, profundizaremos en el estudio del shell más adelante.

Finalmente el sistema operativo UNIX algunas veces parece ser más difícil de lo que es, así mismo es difícil para un principiante saber aprovechar al máximo las facilidades disponibles pero por suerte es fácil comenzar a utilizarlas. El conocimiento de unos cuantos programas como son los comandos básicos es suficiente para dar el primer paso, posteriormente se debe aprender la estructura del sistema operativo, que en si no es complicada pero requiere que se analice a conciencia para comprender la forma de trabajo entre los procesos y de esta manera entender la forma de administración del sistema operativo.

Hay muy pocas cosas que se pueden hacer accidentalmente y que ocasionen problemas a otros usuarios o a nosotros mismos, pues UNIX es un sistema operativo con un elevado nivel de seguridad tanto hacia los usuarios mismos del sistema, los cuales tienen accesos restringidos hacia cualquier elemento que no pertenezca a ellos y hacia usuarios externos que trabajen temporalmente en el servidor, además de proteger en gran medida al sistema de ataques externos por parte de gente extraña que pretenda introducirse sin autorización.

El estudio de este capítulo nos dará un panorama general sobre lo que es el sistema operativo UNIX y nos proporcionará las herramientas necesarias para trabajar en esta plataforma totalmente diferente, de tal manera, comenzaremos a explorar en un mundo diferente llamado UNIX.

3.1.2 Historia y evolución de UNIX

El sistema operativo UNIX se inicio en 1969 en una máquina DEC PDP-7 desechada en los laboratorios Bell. Ken Thompson, con ideas y el apoyo de Rudd Canaday, Doug McIlroy, Joe Ossanna y Dennis Ritchie, escribió un sistema operativo de tiempo compartido y de uso general lo bastante adecuado y cómodo como para atraer a usuarios entusiastas y con suficiente credibilidad para justificar la adquisición de otra máquina más grande para propósitos de continuar con la investigación, dicha máquina era una PDP-11-20. Uno de los primeros usuarios fue Dennis Ritchie, quien ayudó a trasladar el sistema a que había sido desarrollado a la PDP-11 en el año de 1970 [Kernighan, 1987].

Dennis Ritchie diseño y escribió un compilador para el lenguaje de programación C, mismo que fue utilizado más tarde en 1973 para que Ritchie y Thompson reescribieran el kernel de UNIX en el lenguaje de programación C, de ésta manera se rompió la tradición de que el software de sistemas estaba escrito solamente en lenguaje ensamblador. Así el sistema UNIX adquirió la estructura esencial con la cual lo conocemos hoy en día.

En el año de 1974 fue introducido a las universidades para fines educacionales y al cabo de pocos años estaba ya disponible para su uso comercial. En aquel entonces los sistemas UNIX prosperaron en los laboratorios Bell y de allí se difundieron a otros laboratorios, centros de investigación, proyectos de desarrollo de software, centros de procesamiento y sistemas de apoyo a las operaciones en las compañías de teléfonos de los Estados Unidos. Desde entonces cobró gran popularidad y comenzó a utilizarse en muchos lugares en el mundo entero y ha sido instalado en millones de computadoras que van desde micro hasta macrocomputadoras.

De ésta manera podemos distinguir que el éxito del sistema operativo UNIX se debe a varias razones; primero, está escrito en el lenguaje de programación C lo cual ofrece portabilidad hacia otras arquitecturas de computadora, y ofrece a este sistema una gran ventaja comercial. Otra razón es que su código fuente está disponible y escrito en un lenguaje de alto nivel, lo cual permite hacer modificaciones al sistema original y adaptarlo a exigencias particulares. Finalmente algo muy importante, es un excelente sistema operativo especialmente para los programadores. Su ambiente de programación es de extraordinaria riqueza y productividad, pues ofrece grandes facilidades para el desarrollo de sistemas.

Aun cuando UNIX introduce varios programas y métodos innovadores, su eficiencia no se debe a un programa o idea en particular, sino a su enfoque de programación, una nueva filosofía de como utilizar un equipo de cómputo. Aunque ésta filosofía no puede describirse en una sola frase, se basa en la convicción de que un sistema depende más de las relaciones entre los programas que de los programas propiamente dichos. Muchos programas en

UNIX hacen aisladamente tareas triviales, pero al combinarse con otros programas se convierten en herramientas poderosas.

Al irse difundiendo día a día este sistema operativo ha disminuido el número de usuarios expertos en su aplicación, los cuales hace algunos años eran difíciles de encontrar, actualmente UNIX ha ofrecido muchas mejoras y facilidades incluso con la introducción de Linux que es un sistema operativo UNIX de distribución libre; gracias a éste ha sido posible para muchos usuarios transportar el poder de este sistema operativo a PCs y otras plataformas como Macintosh para trabajar de manera independiente o formando redes.

Por último, cabe mencionar que UNIX a pesar de ser uno de los sistemas operativos más viejos, tiene un gran futuro, pues ningún otro sistema ha podido igualar su gran capacidad y su forma de trabajo tan sencilla y tan poderosa, tal vez deberán pasar muchos años para que llegue otro sistema capaz de sustituir a al incomparable UNIX.

3.1.3 Ventajas y desventajas de los sistemas operativos UNIX

3.1.3.1 Ventajas

- UNIX es un sistema multiusuario real, puede correr cualquier aplicación en el servidor. De esta manera pueden trabajar más de un usuario en el servidor, inclusive simultáneamente.

- Es escalable, con soporte para arquitecturas de 64 bits. UNIX es de los pocos sistemas operativos con soporte para 64 bits, con lo cual se vuelve más poderoso y ofrece mayores ventajas de procesamiento.

- El costo de las diferentes variantes de UNIX es muy reducido y algunas son gratis, como FreeBSD y Linux que por su parte constituyen una categoría de sistemas operativos de licencia libre y sin embargo son igualmente confiables ya que cuentan con las mismas características de los sistemas UNIX comerciales, incluso algunas veces ofrecen mayores ventajas.

- Se pueden activar y desactivar drivers o dispositivos sin necesidad de reiniciar el sistema. Esto constituye una de las grandes ventajas, pues podemos agregar o eliminar dispositivos sin tener que dar de baja el sistema para que sean reconocidos.

- UNIX puede trabajar con CLI (Command Line Interface). La interfaz principal desde que UNIX fue creado fue la línea de comando, aunque ahora en día los usuarios de UNIX se pueden dar el lujo de utilizar la interfaz gráfica que más les guste, pues existen versiones de diferentes proveedores.

- Los kernels de UNIX se confeccionan según las necesidades. Así el usuario o programador puede adaptar el sistema a sus necesidades.

- Ofrece la capacidad de realizar cómputo remotamente. Un trabajo dificil para algunos sistemas operativos que utilizan sus propios protocolos, UNIX lo hace de manera transparente a través de TCP/IP.

- Es la mejor solución para enormes bases de datos. Ofrece mayor seguridad del sistema. Por su gran consistencia y por su diseño podemos trabajar en UNIX de una manera confiable sin preocuparnos por la integridad del sistema.

Los sistemas UNIX ofrecen grandes ventajas en comparación con muchos otros sistemas operativos como lo es Windows NT uno de sus más fuertes competidores. UNIX ha desarrollado su propia filosofía de trabajo con la cual se ofrece la mayor parte de sus ventajas. Ésta, ha influido fuertemente sobre la estructura del sistema y la forma de operar.

La filosofia de UNIX se basa en la idea de que un sistema operativo potente y complejo debe ser simple, general y extensible. Con estas premisas UNIX ofrece importantes ventajas tanto para usuarios como para programadores. De tal manera, podemos aún mencionar muchas más ventajas que nos ofrece. Principalmente contempla los archivos de una manera extremadamente simple y general dentro de un modelo único. Trata de la misma manera los directorios, los archivos ordinarios, los dispositivos, tales como impresoras y unidades de almacenamiento, los teclados y terminales de pantalla. El sistema de archivos oculta detalles del hardware subyacente al usuario. Por ejemplo, no necesitamos saber sobre que unidad de disco se encuentra un archivo, esta simplicidad permite concentrarse sobre lo que es realmente importante. En las redes el concepto de sistema de archivos remoto evita la necesidad de conocer sobre que máquina esta almacenado un determinado archivo.

Otra de sus ventajas es que al tratar al teclado y a la pantalla como archivos, también permite utilizar con ellos los mismos programas u órdenes que con archivos almacenados de manera ordinaria, tomando la entrada desde la terminal o visualizando la información sobre ésta. UNIX trata la entrada y salida de una forma muy simple y consistente utilizando entrada estándar y salida estándar, por ejemplo: la entrada a una orden puede tomarse directamente de la terminal o de la salida de otra orden o programa sin necesidad de hacer adaptaciones especiales o versiones diferentes de los programas.

3.1.3.2 Desventajas

En cuanto a las desventajas de UNIX podemos solo señalar muy pocas pues como hemos dicho, UNIX es un sistema operativo sumamente confiable y seguro. Sin embargo vale la pena mencionar algunas de ellas a manera de comparación o para fines de evaluar el sistema.

Como cualquier sistema de computadora, UNIX no es 100% seguro, pues sabemos que no existe algún sistema como tal, sin embargo nos ofrece mayor seguridad que muchos otros. Por lo tanto no tomaremos esto como realmente una desventaja y analizaremos la seguridad más adelante.

- La interfaz de usuario no es muy amistosa en algunas versiones. Aunque UNIX desde sus inicios siempre ha conservado como interfaz básica para el usuario la línea de comando, hoy en día los usuarios son más exigentes y demandan interfaces sumamente sencillas en ambientes gráficos.

- Requiere capacitación, ya que debido a su complejidad, no cualquiera puede usarlo. Algunas veces incluso desde la instalación requiere de conocimientos muy avanzados sobre el equipo y las condiciones en que se instalará.

- Padece la falta de aplicaciones comerciales con nombres importantes. Esto es debido a que la mayor parte de los usuarios aquí en México, estamos acostumbrados a trabajar en ambientes comerciales y con software sumamente comercial, es por esto, que para una empresa implicaría gastos de capacitación el aprender a operar con un software diferente y en una plataforma diferente a las tradicionales y comerciales como Windows.

- Hay discrepancias entre los distintos diseñadores y vendedores de UNIX. Los sistemas UNIX aunque tienen una base en cuanto a su diseño, existen diferencias entre las ediciones pues cada compañía agrega herramientas para complementar y hacer mejor su sistema operativo según su perfil o las necesidades de su mercado.

3.1.4 Variantes de los sistemas UNIX

Cuando finalmente se completo la versión UNIX en los laboratorios de Bell de AT&T, muchas personas creían que sería la versión de UNIX utilizada por todos los fabricantes sobre todas las plataformas. Por una serie de razones esto no ocurrió, muchos fabricantes no quisieron abandonar sus propias versiones del sistema operativo y no querían tampoco utilizar la versión final de AT&T llamada "Sistema V". Incluso muchas variantes de UNIX incluían herramientas y capacidades que no estaban presentes en el Sistema V y tampoco corrían dentro de éste. Como consecuencia de esto, muchas variantes siguen siendo utilizadas, muchas de ellas basadas en la versión final del Sistema V, Estas variantes incluyen versiones de UNIX suministradas por fabricantes de computadoras, diseñadas para correr sobre sus propias máquinas, tales como AIX de IBM, HP-UX de Hewlett Packard e IRIX de Silicon Graphics. Otras variantes están diseñadas para correr sobre plataformas estándar de la industria, tales como SCO UNIX, UnixWare, Solaris y Linux.
Describiremos brevemente algunas de las variantes del sistema UNIX.

3.1.4.1 El sistema solaris

Solaris es el sistema operativo original de Sun Microsystems, es desarrollado por una empresa llamada SunSoft la cual es una subsidiaria de desarrollo y distribución de software creada por Sun Microsystems.

Inicialmente este sistema operativo se llamó SunOS y fue basado en el sistema operativo creado por los laboratorios Bell de AT&T. Existen actualmente versiones del sistema Solaris para estaciones Intel y SPARC. También se está trabajando en una implementación para los procesadores PowerPC. Existen versiones de Solaris independientes de servidor,

hay también un amplio rango de herramientas de desarrollo tanto en ambiente gráfico como caracter, Solaris tiene un mayor soporte para la interfaz gráfica MOTIF que otras ediciones de UNIX.

La versión actual de Solaris es la 7, es un completo sistema de 64 bits que ofrece interoperabilidad reforzada, instalación, administración y configuración muy sencilla. Mantiene una completa compatibilidad binaria entre las versiones de Solaris 2.x, con los procesadores Intel y SPARCTM.

3.1.4.2 UNIXWARE

El nombre dado al producto desarrollado por Novell para correr sus productos basados en UNIX es UNIXWARE, este sistema operativo cumple con los estándares POSIX establecidos para los productos UNIX, sobretodo en lo que se refiere a aplicaciones gráficas.

Existen dos ediciones del sistema, una versión está diseñada para trabajar de manera independiente o de sobremesa llamada SCO UNIXWARE, otra versión es el SCO Open Server, diseñado para utilizarse como servidor. Este sistema operativo sólo está disponible para las plataformas Intel con arquitectura CISC pero se está portando hacia computadoras basadas en la arquitectura RISC, UNIXWARE cuenta también con soporte para aplicaciones gráficas puesto que integra la interfaz gráfica MOTIF.

La última versión en el mercado de este sistema es la 7.1, esta incluye soporte Plug and Play para reconocimiento de dispositivos agregados al sistema, lo cual hace más fácil su instalación para aquellos usuarios que no son tan experimentados. Además incluye soporte para correr sesiones de Windows 9x, Windows 3.x y DOS bajo el sistema operativo. Estas son solo algunas de las características sobresalientes de UNIXWARE.

3.1.4.3 HP-UX

Hewlett Packard también desarrolló su propio sistema operativo UNIX para sus equipos y estaciones de trabajo diseñados para trabajar en este ambiente. La edición de HP se llama HP-UX y está basado en el mismo esquema de trabajo que todos los sistemas UNIX solo que incluye muchas herramientas propias de HP para mejorar y ayudar a las tareas cotidianas, sobretodo a las administrativas. La versión más reciente de HP-UX es HP-UX 11.00 lanzada en 1998 y ésta ha sido mejorada y adaptada a conforme a los estándares de UNIX.

Con aproximadamente 15,000 aplicaciones que corren y están diseñadas para HP-UX, ésta es una de las plataformas preferidas de industria para desarrollo de aplicaciones y despliegue de soluciones de misión crítica, opera en un completo ambiente de 64 bits.

3.1.4.4 IRIX

IRIX es la versión de UNIX que pertenece a Silicon Graphics para ser utilizada en sus equipos basados en procesadores con arquitectura RISC. IRIX a diferencia de otras

implementaciones, es un sistema operativo de 64 bits con lo cual optimiza su rendimiento y mejora el trabajo en aplicaciones gráficas que requieren un procesamiento mejor del CPU. Realmente IRIX fue diseñado para trabajar bajo el esquema de diseño gráfico y es por eso que su especialidad está ahí

3.1.4.5 DEC OSF/1

Digital Equipment Corporation también ha desarrollado su propia edición de UNIX y ha vendido una gran cantidad de equipos en los que se ejecuta una versión de UNIX llamado ULTRIX.

Sin embargo con la llegada de los nuevos procesadores CISC Alpha, DEC se ha enfocado al desarrollo de una nueva edición de su sistema operativo llamado DEC OSF/1. Ese nuevo desarrollo cuenta con soporte para 64 bits, soporte en tiempo real, gestión de memoria incrementada, multiprocesamiento simétrico, y rápido restablecimiento del sistema de archivos en caso de fallo. Este sistema está basado en el diseño del sistema de los laboratorios Bell de At&T y BSD de la Universidad de Berkeley.

3.1.4.6 LINUX

Linux es otra de las variantes de UNIX, aunque este sistema no es desarrollado por alguna empresa en particular, es un sistema que está cobrando mucha fuerza en el mercado de los sistemas operativos.

Fue iniciado por Linus Torvalds en 1991 cuando era estudiante de la universidad de Helsinki en Finlandia, actualmente miles de personas distribuidas en el mundo entero colaboran en su desarrollo.

Originalmente este sistema fue desarrollado para trabajar solamente en PCs basadas en los procesadores x86 pero en la actualidad existen versiones de Linux para una amplia gama de procesadores y arquitecturas. Linux es un sistema no comercial, lo contrario de los sistemas tradicionales UNIX, posee un Copyright bajo los términos de licencia pública general GNU para prevenir que pueda ser vendido y distribuido sin permitir al comprador que lo copie libremente.

Existen miles de aplicaciones que corren bajo Linux puesto que ha recibido gran aceptación en el mercado y muchas empresas desarrolladoras de software han cooperado desarrollando aplicaciones que incluso también son libres de licencia. Un ejemplo de ello es la empresa Oracle que ha creado y distribuido sus aplicaciones para desarrollo de bases de datos sin costo alguno para los usuarios de Linux.

3.1.4.7 A/UX

La empresa Apple cuenta con una versión de UNIX llamada A/UX lo cual es una combinación del sistema operativo de Macintosh System 7 con UNIX, por lo cual posee la completa funcionalidad del sistema de Macintosh y el poder de UNIX en un mismo paquete. Este sistema está basado en el diseño del sistema operativo UNIX de Berkeley

mas muchas otras características tomadas del sistema desarrollado por los laboratorios Bell de At&T. Casi todas las aplicaciones Macintosh pueden usarse bajo A/UX y muchas aplicaciones nativas de UNIX pueden ser portadas a este sistema. A/UX está diseñado para operar con procesadores Motorola 680x0. Cabe mencionar que no es posible ejecutarlo en computadoras con procesadores PowerPC.

3.1.4.8 AIX

AIX es una versión del sistema operativo UNIX de IBM. Fue desarrollado exclusivamente para ejecutarse con procesadores POWER y PowerPC en estaciones IBM.AIX integra características de los sistemas operativos UNIX de Bell y Berkeley.

La versión actual de AIX es la 4.3. AIX incluye soporte para la interfaz gráfica MOTIF, la interfaz gráfica de usuario desarrollada por OSF (Open Systems Fundation) e incluye una implementación del ambiente de escritorio común (Common Desktop Environment). Cuenta con más de 10.000 aplicaciones que están listas para ejecutarse bajo AIX.

3.1.5 Usuarios del sistema operativo UNIX

Los usuarios en UNIX están divididos en grupos, asignación que normalmente se lleva a cabo por el Administrador del sistema. El superusurio o administrador del sistema, tiene la responsabilidad de administrar. Esta tarea incluye darles de alta para que puedan entrar en el sistema, establecer privilegios, crear y asignar directorios, asignar usuarios a grupos y darles de baja cuando sea preciso.

Cada usuario deberá de tener un nombre de entrada único. Con ello se le podrá identificar y evitará que un usuario borre o modifique archivos de otros. Cada usuario debe de tener una contraseña. La única excepción a esta regla es cuando sólo hay un usuario que puede acceder al sistema y éste no tiene ninguna conexión por medio de módem o red con otras computadoras.

Es importante que el administrador del sistema lleve un control sobre las cuentas que deben estar vigentes, y cuando ya no hay razón para que una persona entre al sistema, hay que asegurarse de que no pueda hacerlo. Deberá eliminarse su nombre de entrada junto con los archivos que el resto de los usuarios no necesite y su directorio de trabajo.

Existen diferentes tipos de usuarios en un sistema UNIX. Para ello cada usuario es miembro de un grupo, pueden darse distintas posibilidades o privilegios a distintos tipos de grupos. Por ejemplo, parece más lógico dar acceso a distintos tipos de archivos a un grupo que utiliza el sistema para analizar las ventas de la compañía que a otro cuya tarea principal es la investigación de nuevos productos.

El hecho de que se organice a los usuarios por grupo ofrece la ventaja de asignar ciertos privilegios propios para ese grupo, ya sea para tener acceso a algunos programas exclusivos o tener derechos sobre ciertos archivos o directorios.

3.1.5.1 Identificación de un usuario en el sistema

Cuando se da de alta un usuario, el resultado es una entrada en el archivo de contraseñas de usuarios ubicado en */etc/passwd* el cual puede ser editado. El archivo *passwd* contiene las siguientes entradas para identificar a un usuario:

Campo	Descripción
Login_name	El nombre que se utiliza para entrar.
Password	Contraseña para identificar al usuario.
User_ID	Un nombre o número para identificar en el sistema al usuario.
Group_ID	Un nombre único de grupo para identificar el grupo primario al que pertenece el usuario.
User_information	Descripción del usuario como: Nombre, Teléfono, Puesto, etc.
Login_directory	Directorio Inicial del usuario o directorio home
Login_shell	El shell utilizado por el usuario al entrar

Tabla 3-1 campos de entrada del archivo /etc/passwd

Para agregar usuarios al sistema se pueden utilizar diferentes herramientas de administración o simplemente llevar a cabo el proceso creando las entradas necesarias en los archivos de configuración. Para esto, cuando se asigna una nueva cuenta a un usuario, esta debe constar de las siguientes partes:

1. Un nombre de usuario (login-name) o identificador y una contraseña (password) secreta para corroborar dicha identificación.
2. UID y GID, estos números indican el número de usuario asignado y el número del grupo al cual pertenece.
3. Un directorio base, el cual es un directorio en el sistema en donde el usuario va a tener su información y trabajos, por tanto este usuario tendrá derecho a escribir, leer y ejecutar programas en su directorio.
4. Un interprete de comandos, shell, que el usuario personalmente utilizará para darle instrucciones al núcleo o kernel; el shell existe ya que el usuario necesita un traductor para entrar en contacto directo con el kernel.

Las contraseñas se definen utilizando el comando "passwd". El administrador debe definir una contraseña para cada usuario que se añada al sistema. Los usuarios tienen la opción de cambiar esta contraseña cuando entran al sistema. Realmente esto debe ser una practica constante o bien una costumbre que asegure en gran medida que un intruso no podrá entrar al sistema por medio de la cuenta de alguno de los usuarios.

Para crear un grupo si no se tiene una herramienta de administración, es necesario editar el archivo */etc/group* y escribir la información correspondiente a este grupo nuevo. En el archivo */etc/group*, cada grupo tiene asignado un identificador de grupo exclusivo, de hecho el sistema busca el identificador de grupo y no su nombre, por eso es fundamental no asignar el mismo número a más de un grupo.

3.1.5.2 Directorios de usuario

Es muy importante agrupar los directorios de usuario de una forma lógica, especialmente si se tiene planeado tener muchos usuarios en sistema. En general se debe procurar poner todos los directorios de usuario en una misma máquina y bajo un mismo directorio de alto nivel. De esta forma, pueden agruparse de acuerdo con los criterios que se tomaron en cuenta para crear dichos grupos.

Por ejemplo, se puede especificar que */home* sea el directorio de alto nivel para los directorios de usuario, bajo */home* se pueden agrupar los usuarios por departamentos o áreas. Los usuarios de ventas tendrían cuentas bajo */home/ventas*, desarrollo estaría bajo */home/desarrollo* y así sucesivamente. Puesto que los directorios de usuario pueden utilizar mucho espacio en disco, este sistema permite además situar grupos lógicos de usuarios en distintos sistemas físicos. Según se vaya necesitando más espacio, se pueden crear otras categorías para directorios de usuarios y montarlas en otro sistema de archivos utilizando como punto de montaje el directorio */home*.

3.1.6 Configuración del sistema UNIX

Antes de que se pueda ver el indicador shell al entrar a una sesión UNIX, este configura el entorno de forma predeterminada. El entorno UNIX contiene configuraciones y datos que controlan la sesión mientras permanecemos conectados al sistema. Al igual que muchas otras cosas en UNIX, somos libres de cambiar los parámetros de configuración de acuerdo a nuestras necesidades.

El entorno de sesión se encuentra dividido en dos componentes:

El primero es llamado entorno de terminal, controla la terminal (más concretamente, el comportamiento del puerto de la computadora al que se conecta el cable procedente de la terminal).

El segundo componente, llamado entorno de shell, controla varios aspectos del shell y de los programas que ejecuta.

3.1.6.1 Configuración de la terminal

La sesión de entrada al sistema consta de dos programas independientes que se ejecutan en paralelo para proporcionar la apariencia de estar utilizando una máquina en exclusiva. Aunque el shell es el programa que recibe las instrucciones y las ejecuta, antes de que el shell conozca las órdenes, todo lo que escribimos pasa primero por el programa relativamente transparente llamado *controlador de dispositivos*.

El controlador de dispositivos controla la terminal, recibe los caracteres que escribimos y determina lo que ha de hacer con ellos, si hay algo que hacer antes de pasarlos al shell para que los interprete. Igualmente, cada carácter generado por el shell debe pasar por el controlador de dispositivo antes de ser trasladado a la terminal.

UNIX es el único sistema en que, para un programa, todos los dispositivos conectados al sistema son muy similares entre ellos y todos los dispositivos parecen archivos. Los distintos controladores de dispositivos del sistema se ocupan de realizar esta transformación. Un disco duro del sistema se comporta de manera muy distinta a la de una terminal; sin embargo, el trabajo de los respectivos controladores permite que para un programa resulten idénticos.

Puesto que la terminal se encuentra siempre conectada al sistema, el controlador de dispositivos le permite definir caracteres especiales, llamados *caracteres de control*, que sirven como marcadores de final de archivo y final de línea para el shell. El controlador de dispositivos también permite definir caracteres de control que envían señales a un proceso de ejecución (como la señal de interrupción que puede, en la mayoría de los casos, detener un proceso en ejecución y devolvernos al shell). La figura 3-2 muestra la forma de comportamiento entre el kernel, el shell y el controlador de dispositivos de UNIX.

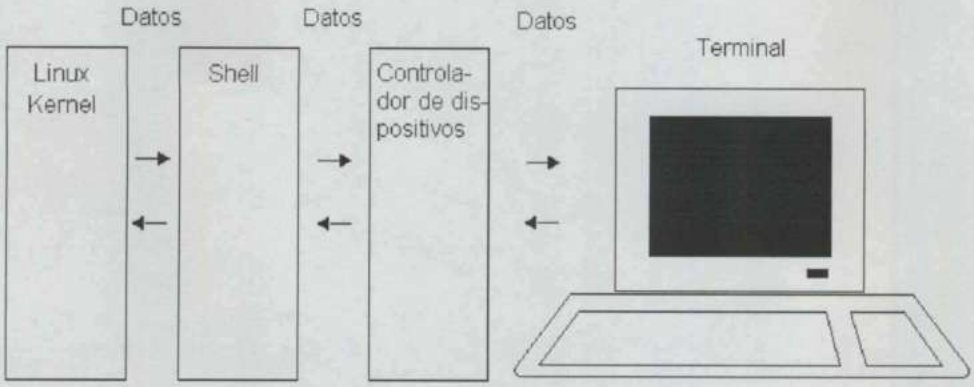

Figura 3-2 Muestra la forma en que UNIX interactúa con el usuario mediante la orden shell.

Pueden configurarse docenas de parámetros para una terminal, aunque la mayoría de ellos se gestionan de forma diferente. Existen, sin embargo, unos cuantos parámetros y modalidades sobre los que debemos estar informados.

El controlador de dispositivo tiene modalidades de operación llamadas tratada y primitiva [Moritsugu, 1998]. En la modalidad primitiva, todos los caracteres que se escriben pasan directamente al shell o a un programa ejecutado por el shell. Los programas como editores y hojas de cálculo, necesitan la modalidad primitiva y la configuran automáticamente. Cuando estos programas terminan, normalmente vuelven a restaurar su terminal a la modalidad tratada, aunque no siempre ocurre así. Cuando la terminal se encuentra en la modalidad primitiva, ésta no responde a teclas de control como por ejemplo la tecla de interrupción.

Cuando la terminal se encuentra en la modalidad tratada, el controlador de dispositivos interpreta cada tecla que pulsamos. Las teclas normales se almacenan en una memoria intermedia hasta que se pulsa la tecla de final de línea, que normalmente corresponde a la

tecla <Enter> o <Return> (sin embargo puede cambiarse). Cuando el controlador de dispositivos recibe el carácter de final de línea, interpreta toda la línea antes de transferirla al shell o programa de aplicación.

La orden que se utiliza para configurar y mostrar estos parámetros es *stty*. La orden *stty* es una abreviación de *set teletype*, es decir, configurar el teletipo. En otros tiempos de la informática sólo se disponía de un terminal de teletipo y por ello gran parte de la terminología UNIX procede de aquella época [Hahn, 1995]. Por ejemplo para mostrar todas las configuraciones actuales de la terminal en la que trabajamos, basta con escribir la orden *stty –a* desde la línea de órdenes y presionar <Enter>.

Aunque se pueden establecer todas las configuraciones que se listan con el comando anterior, por cuestiones prácticas, los usuarios, normalmente, sólo restauran las teclas de interrupción y finalizar. Por ejemplo si deseáramos cambiar la tecla finalizar de ^u a ^c deberemos teclear:

```
stty kill '^c'
```

3.1.6.2 Configuración del shell

Parte de la conexión al sistema, es decir, crear una sesión UNIX, consiste en la creación de su entorno. Todos los procesos UNIX tienen su propio entorno, que es independiente y distinto del propio programa. Podría decirse que un programa se ejecuta desde dentro de un entorno. El entorno UNIX, llamado el entorno shell, consiste en un número de variables y sus valores, que permiten a un programa en ejecución, como un shell, determinar el aspecto que tiene el entorno.

El entorno hace referencia a aspectos como el nombre del shell, directorio de usuario y el tipo de terminal utilizado. Muchas de estas variables se definan durante el proceso de la entrada al sistema y no deben modificarse. Pueden añadirse variables o modificarse, siempre y cuando no estén marcadas como solo de lectura.

Las variables se configuran en el entorno con el formato VARIABLE=VALOR. El significado de variable puede ser cualquier cosa que se desee. Sin embargo muchas variables tienen significados predeterminados para muchos programas UNIX estándares. Por ejemplo, la variable *TERM* se define como el nombre de tipo de terminal especificado en una de las bases de datos estándares de terminal UNIX.

Digital Equipment Corporation fabricó un terminal muy popular conocido como vt-100. Las características de este terminal han sido copiadas por muchos otros fabricantes y han sido a menudo emuladas en software para computadoras personales; se representa en el entorno UNIX como *TERM=vt100.*

En el entorno existen muchas otras variables predeterminadas. Si utilizamos el shell C podemos listar estas variables utilizando la orden *printenv*; para los shells Bourne o Korn se utiliza la orden *set*.

A continuación se presenta una lista de las variables de entorno más comunes y su utilización. Estas variables deben ser configuradas por el usuario para personalizar su entorno de trabajo dentro de una sesión UNIX.

Variable	Descripción
HOME=/home/login	Configura el directorio de usuario, que es la localización desde inicio de sesión para un usuario.
LOGNAME=login	LOGNAME se configura automáticamente, igual que el identificador de entrada al sistema.
PATH=path	La variable PATH representa la lista de directorios que el shell examina para ordenes.
PS1=prompt	PS1 es el indicador primario de shell que define el aspecto que tiene el indicador. Por default es ($).
PWD=directory	PWD se configura automáticamente. Esta variable indica el lugar en el que se encuentra el usuario en el sistema de archivos.
SHELL=shell	SHELL identifica la localización del programa que sirve como shell. Se puede configurar en el archivo .profile o .login
TERM=termtype	Establece el nombre del tipo de terminal, tal y como se especifica por la base de datos del terminal.

Tabla 3-2 variables de entorno más comunes y su descripción

3.1.7 Acceso a un sistema UNIX

3.1.7.1 Acceso a un sistema UNIX desde una PC

Si no somos administradores o no tenemos acceso al servidor UNIX o alguna terminal X para UNIX, entonces probablemente lo más seguro será que utilizamos una PC para emular una terminal y abrir la sesión en el sistema. Existen muchos paquetes de aplicación, llamados *emuladores de terminal*, que se ejecutan en una PC y hacen posible la conexión a un sistema UNIX. Todos los emuladores de terminal funcionan básicamente de la misma forma, actuando como si se tratara de un terminal conectado directamente a una máquina UNIX. Esto nos permite introducir órdenes de la misma manera en que lo haríamos si estuviésemos utilizando una auténtica terminal. Algunos emuladores permiten ejecutar al mismo tiempo órdenes DOS mientras se sigue utilizando el sistema UNIX. Existe una variedad de estos emuladores, tanto para DOS como para Windows, muchos de ellos permiten hacer transferencia de archivos entre la PC y el sistema UNIX.

3.1.7.2 Entrada al sistema

Cuando un usuario se introduce por primera vez a la red se le otorga su identificador y se le asigna además una contraseña para que pueda obtener un mayor grado de seguridad y privacidad en su cuenta, la contraseña debe ser conocida sólo por el dueño de la cuenta y debe cambiarse periódicamente.

A continuación se muestra cómo aparece el procedimiento de entrada en la pantalla de la terminal cuando se accesa a un sistema UNIX.

```
sunserver% telnet 148.202.3.5
Trying 148.202.3.5...
Connected to foreigner.
Escape character is '^['

SunOS UNIX (foreigner)

login: usuario
Password:
```

Este procedimiento puede verse diferente. Si en la pantalla de la terminal no aparece la palabra *login*: debe comprobarse si está conectada y oprimir la tecla <Enter> varias veces. Si la máquina no despliega nada debemos consultar al administrador del sistema.

La primera línea de la pantalla muestra el indicador *login*: de UNIX seguido por la respuesta del usuario. El usuario introdujo *usuario* que es su nombre login y luego presiono <Enter>. Debemos entrar al sistema asegurándonos de teclear el nombre de usuario exactamente como fue asignado. La rutina que verifica el nombre de usuario y la contraseña es sensible a letras mayúsculas y minúsculas.

La segunda línea de la pantalla muestra el indicador *password*:. Si la cuenta no requiere de uno, no se verá el indicador. En el ejemplo, el usuario respondió el indicador *password*: con su contraseña seguida de <Enter>. Como medida de seguridad, UNIX nunca desplegará en pantalla lo tecleado como contraseña.

Al lograr entrar al sistema en forma correcta aparece un mensaje y un indicador. El mensaje puede ser información sobre su última entrada al sistema o información a los usuarios de parte de los administradores.

En C Shell, el indicador "prompt" suele ser el nombre de la máquina seguido por un signo de porcentaje. Este indicador significa que el sistema está preparado para recibir una instrucción.

Cuando se introduce un nombre de usuario o una contraseña no válidos, en la pantalla se despliega el mensaje siguiente: `login incorrect`

Este mensaje indica que el nombre del usuario o la contraseña se introdujeron en forma incorrecta o que no son válidos. El mensaje no distingue entre un nombre de usuario inaceptable y una contraseña inaceptable.

Una vez dentro del sistema, se está en comunicación con el interprete de comandos llamado shell. El shell tiene una parte importante en toda la comunicación con el sistema operativo UNIX. Cuando se introduce un comando en la terminal (como respuesta al indicador shell), el shell lo interpreta e inicia la acción apropiada. Estos comandos se teclean después de que aparece el indicador de comandos.

3.1.7.3 Cambio de contraseña

Como ya se mencionó anteriormente, para entrar a un sistema UNIX es necesario contar con un nombre de usuario y una contraseña que son asignados por el administrador, es prudente cambiar la contraseña con regularidad y no decirlo a nadie para que el usuario pueda mantener un cierto grado de seguridad en sus archivos. Una buena manera de crear una contraseña es combinar letras mayúsculas y minúsculas con números de una longitud aproximada de 8 caracteres. El comando *passwd* sirve para cambiar la contraseña actual por una nueva (para mayor seguridad del usuario la contraseña tecleada no aparece en la pantalla). Los pasos que se siguen se mencionan a continuación.

sunserver% passwd	Se teclea el comando y se presiona <Enter>
Changing NIS password for <login>	
Old password:	Aquí se debe teclear la contraseña actual
Type new password	enseguida se teclea la nueva contraseña
Retype password	El sistema pide que se vuelva a introducir la contraseña

En caso de que nos equivoquemos al volver a escribir la nueva contraseña tecleando algo diferente, se desplegará el siguiente mensaje de error:

```
Mismatch – password unchanged
```

Y la contraseña no será cambiada. En caso contrario, lo que se introduzca será la contraseña para entrar al sistema y debemos recordarla para siguientes accesos al sistema.

3.1.7.4 Salida del sistema

Cuando terminemos una sesión de trabajo, podemos salir escribiendo la palabra *logout* en el indicador de comandos como se muestra enseguida:

```
sunserver% logout
```

En el ejemplo, *sunserver%* es el indicador; al teclear *logout* salimos del sistema y en la pantalla aparece de nuevo el indicador *login*:

3.1.8 Archivos y directorios

3.1.8.1 El sistema de archivos

Bajo UNIX, el espacio de archivos que resulta visible para los usuarios se basa en una estructura en árbol, con la raíz en la parte superior. Los distintos directorios y archivos se ramifican hacia abajo desde la raíz. El directorio superior, "/", se conoce como el directorio raíz. La figura 3-3 muestra la estructura de árbol de un sistema de archivos.

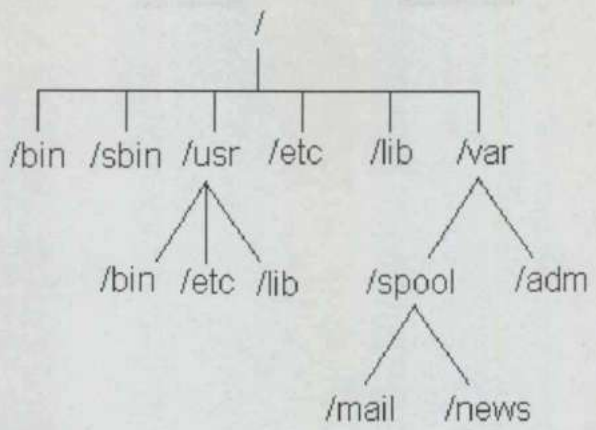

Figura 3-3 Una sección del sistema de archivos UNIX representado como árbol vuelto de revés, con la raíz arriba y las ramas y hojas cayendo hacia abajo.

Desde el punto de vista del usuario, éste árbol parece una entidad uniforme; lo único que se ve son directorios y archivos. En realidad, muchos de los directorios que se ven en el árbol componen distintas particiones del disco, situadas en diferentes discos e incluso, en distintas computadoras. Cuando una de esas particiones de disco se anexa al árbol en un directorio conocido como punto de montaje, éste y todos los directorios por debajo de él se denominan un *sistema de archivos*.

El sistema operativo UNIX se compone de varios directorios y de muchos archivos distintos. Según se hayan seleccionando durante el proceso de instalación, esos directorios pueden ser distintos sistemas de archivos. Normalmente, la mayor parte del sistema operativo reside en dos sistemas de archivos: el sistema de archivos raíz, conocido como /, y el sistema de archivos montado bajo /usr, llamado *user*.

Si nos cambiamos al directorio raíz con la orden cd / y se pide un listado del directorio, se ven varios directorios. Estos directorios son los que componen el sistema de archivos raíz y también proporcionan los puntos de montaje para otros sistemas de archivos.

El directorio /bin tiene programas ejecutables, conocidos como binarios. De hecho bin es la abreviatura de binario. Estos programas son archivos esenciales para el sistema y muchos de las órdenes, como *ls*, son en realidad programas que están en este directorio.

El directorio /sbin también se utiliza para almacenar archivos binarios del sistema. La mayoría de archivos de este directorio se utilizan para administrar el sistema.

El directorio /etc es muy importante y contiene muchos de los archivos de configuración del sistema. Esencialmente, estos son los archivos que conforman la personalidad de un sistema UNIX. Aquí también se encuentra el archivo de contraseñas, *passwd*, así como la lista de los sistemas de archivos que se montarán al iniciar el sistema, *fstab*. Además este directorio contiene archivos de inicio para UNIX, la lista de sistemas principales con

direcciones IP que se quieran registrar permanentemente y muchos otros tipos de información de configuración.

El directorio /dev contiene archivos especiales que se conocen como archivos de dispositivo. Estos archivos se utilizan para acceder a los distintos tipos de hardware que hay en el sistema.

El directorio /proc es realmente un sistema de archivos virtual. Se utiliza para leer información de los procesos desde la memoria.

El directorio/home es el directorio base de los directorios iniciales de los usuarios. Es normal montarlo como un sistema de archivos separado de forma que los usuarios puedan tener espacio suficiente para sus archivos.

El directorio /var contiene archivos que tienden a cambiar de tamaño con el tiempo. Normalmente varios registros del sistema están ubicados bajo este directorio. El directorio /var/spool y sus subdirectorios se utilizan para almacenar datos temporales sobre algunos servicios.

El directorio /usr y sus subdirectorios son muy importante para el funcionamiento del sistema UNIX. Este directorio contiene varios subdirectorios, que a su vez, contienen los programas más importantes en el sistema. Normalmente los subdirectorios de /usr contienen los grandes paquetes que se instalan, por lo tanto normalmente /usr siempre se monta como un sistema de archivos separado.

El sistema UNIX es bastante grande como para mencionar cada unos de sus directorios, subdirectorios y la función de cada uno de ellos, sin embargo en este momento ya podemos tener una idea de lo que es el sistema de archivos. Ahora vamos a pasar a analizar los tipos de archivos que pueden ser identificados dentro de un sistema de archivos en UNIX.

3.1.8.2 Tipos de archivos en UNIX

Existen cuatro tipos básicos de archivos: archivos normales, directorios, enlaces y archivos especiales. Hay varias clases de archivos normales, enlaces y archivos especiales, y un gran número de directorios estándares.

Se puede utilizar la orden *file* para determinar el tipo de archivo. *file* reconoce si el archivo es ejecutable, de texto, de datos y demás tipos. Muchas órdenes de UNIX son sólo secuencias de shell o programas interpretados de forma similar a los archivos por lotes de DOS. *file* puede también utilizarse para saber si una orden UNIX es un programa binario ejecutable o simplemente una secuencia shell.

3.1.8.2.1 Archivos Normales

Los archivos normales son con los que se trabaja la mayor parte del tiempo. Los archivos normales pueden contener texto, código fuente en lenguaje C, archivos de órdenes shell, programas binarios ejecutables y datos de naturaleza diversa. Para UNIX un archivo no es

más que un archivo. La única diferencia es que UNIX sabe cuáles de sus archivos están marcados como ejecutables.

3.1.8.2.2 Archivos de Directorio

Los directorios son archivos que contienen los nombres de archivos y subdirectorios, así como punteros hacia esos archivos y subdirectorios. Los archivos de directorio son el único sitio donde UNIX almacena nombres de archivos. Cuando se lista el contenido de un directorio con la orden *ls*, lo único que se hace en realidad es listar el contenido del archivo de directorio.

Cuando se cambia el nombre de un archivo con la orden *mv* y este está en el directorio actual, lo que en realidad se está haciendo es cambiar la entrada en un archivo de directorio. Si se mueve un archivo desde un directorio a otro, lo único que se está haciendo es mover la descripción del archivo, siempre que naturalmente, el nuevo directorio esté en la misma partición o en el mismo disco fijo.

3.1.8.2.3 Enlaces

Los enlaces normalmente no son archivos, sino simplemente entradas de directorio que señalan al mismo inode de un archivo. La tabla de inodes sigue la pista de todos los enlaces asociados a un archivo, y solo cuando se suprime la última entrada de directorio se deja libre el inode para situarlo en un conjunto de inodes disponibles. Naturalmente los enlaces ordinarios no pueden ir más allá de los límites del dispositivo porque todas las referencias de directorio señalan al mismo inode.

UNIX en la mayoría de sus versiones modernas tiene otro tipo de enlace llamado enlace simbólico, donde la entrada de directorio contiene la entrada de un archivo que en sí mismo es una referencia a otro archivo que está ubicado en otro sitio en el sistema de archivos lógico de UNIX. Un enlace simbólico puede señalar a otro archivo o directorio en un mismo disco, en otro disco o a un archivo o directorio en otra computadora. Una diferencia importante entre un enlace normal y un enlace simbólico es que con los enlaces normales cada uno tiene la misma categoría (es decir, el sistema trata cada enlace como si fuera el archivo original) y no se suprimen los datos hasta que no se suprime el último enlace de ese archivo. Con los enlaces simbólicos cuando se suprime el archivo original, también se suprimen los enlaces simbólicos a ese archivo y, además, estos enlaces no tienen la misma categoría que el original.

Se puede saber si un archivo es un enlace con la orden *ls –l*, porque la respuesta que se obtiene muestra tanto el nombre del archivo local como una indicación del archivo enlazado, como se muestra a continuación:

```
lrwxrwxrwx 1 root root 4 Oct 17 15:27 Info ->info/
```

Los marcadores de permiso de archivo también comienzan con un l para indicar que es un archivo enlazado.

3.1.8.2.4 Archivos especiales

En el sistema de archivos se representan todos los dispositivos físicos asociados a UNIX, incluyendo discos, terminales e impresoras. La mayoría de los dispositivos están ubicados en el directorio /dev. Por ejemplo, si se está trabajando con la consola del sistema, el nombre asociado de dispositivo es /dev/console. Si se está trabajando en una terminal estándar, el nombre del dispositivo puede ser /dev/tty01. etc.

Las terminales e impresoras se denominan "dispositivos especiales por caracteres" pueden aceptar o producir una cadena de caracteres. Por otro lado los discos almacenan datos en bloques que están direccionados por cilindro y sector. En un disco no se puede acceder a un sólo caracter, sino que hay que leer y escribir bloques completos. Esto mismo sucede también con las cintas magnéticas. A este tipo de dispositivos se les denomina "dispositivos especiales por bloques".

Existe al menos otro tipo de dispositivo especial: un FIFO (memoria intermedia first-in-first-out). Los FIFO parecen archivos normales; si se escribe en ellos aumentan de tamaño, pero cuando se leen disminuyen. Se utilizan principalmente en procesos del sistema para que muchos programas puedan enviar información sólo a un proceso de control. Por ejemplo, cuando se imprime un archivo con la orden *lp*, ésta establece el proceso de impresión y señala el proceso "daemon" *lpsched* enviando un mensaje a FIFO. Un proceso daemon, a veces llamado demonio, es un proceso del sistema que actúa sin que el usuario intervenga.

Existe un archivo de dispositivo especial muy útil: /dev/null. Todo lo que se envíe a /dev /null se ignora, algo muy útil cuando no se quiere ver la salida de una orden. Por ejemplo, si no se quieren ver los informes del diagnóstico impresos en el dispositivo estándar de error, se pueden poner en /dev/null/, utilizando la orden siguiente:

```
ls –la> /dev/null
```

3.1.8.3 Permisos de los Archivos

En UNIX, los permisos de los archivos implican algo más que simplemente conocer los permisos que tiene un archivo o directorio para un usuario. Aunque los permisos deciden quién puede leer, escribir o ejecutar un archivo, también deciden el tipo de archivo y la forma de ejecutarlo.

Pueden mostrarse los permisos de un archivo utilizando la forma larga de la orden de listado, *ls –l*. El indicador *–l* señala a la orden *ls* que utilice el listado largo. De esta manera se puede ver un listado de directorio similar al que se muestra a continuación:

```
-rw-rwxr--      1 wmaster admon  7950  Feb 14  10:24 uso_net.iw
-rw-r--r--      1 wmaster admon  8081  Feb 14  13:07 propuesta_m.iw
drw-r--r--      1 wmaster admon  7474  Feb 14  13:07 avanzado
-rw-r--r--      1 wmaster admon 10946 Feb 14  13:50 Propuesta.iw
drw-r--r--      1 wmaster admon  5244  Feb 15  09:17 redes_comer
```

Este listado muestra prácticamente todo lo que se puede saber acerca del archivo desde la entrada del directorio y el inode correspondiente. La primera columna muestra los permisos del archivo, la segunda muestra el número de enlaces a un archivo y la tercera muestra el propietario del archivo. En UNIX el concepto de propiedad tiene tres posibilidades: el propietario, el grupo del propietario y todos los demás. La cuarta columna muestra el grupo al que pertenece el archivo. La quinta muestra el número de bytes en el archivo, la sexta es la fecha y hora de creación y la séptima muestra el nombre del archivo.

La columna de permisos (la primera) se divide en cuatro subcampos:

- rwx rwx rwx

El primer subcampo define el tipo de archivo. Un archivo normal tiene un guión (-) como espacio de reserva; los directorios se marcan con una (d), los archivos especiales por bloque con una (b), los archivos especiales por caracteres con una (c) y los enlaces simbólicos con una (l).

Los tres subcampos siguientes muestran los permisos de lectura, escritura y ejecución del archivo. Por ejemplo, rwx en el primero de estos subcampos significa que el archivo tiene permisos para el propietario de lectura, escritura y ejecución. El segundo subcampo muestra la misma información para la propiedad del grupo del archivo; el tercer subcampo muestra los permisos permitidos para todos los demás usuarios.

Estos campos de permisos pueden mostrar más información; de hecho, existen varios atributos empaquetados en estos tres campos. Por desgracia, el significado de esos atributos depende de la versión de UNIX que se esté utilizando y si el archivo es o no ejecutable.

En estos campos también se puede establecer el *bit adosado*. Este bit indica al sistema que guarde una copia del programa en ejecución después de que termine. Si el programa se ejecuta con frecuencia, el bit adosado puede ahorrar tiempo al sistema porque el programa no tiene que arrancarse de nuevo en memoria desde el disco cada vez que alguien lo ejecuta.

Los permisos de un archivo pueden modificarse con la orden *chmod* que tiene dos sintaxis distintas, una absoluta y una relativa, sin embargo solo veremos la absoluta por ser la que utilizaremos normalmente durante la práctica. Con *chmod* podemos cambiar el modo de acceso a uno o más archivos. Solo el dueño o el superusuario pueden cambiar los permisos a un archivo. Para saber qué permisos de acceso se tiene basta ejecutar *ls –l <archivo>*.

Ya sabiendo que permisos necesitamos establecer, podemos hacerlo especificando tres números octales a la orden *chmod*, Los permisos se calculan sumando los tres valores octales:

4 Permiso de lectura (**r**)
2 Permiso de escritura (**w**)
1 Permiso de ejecución (**x**)

De tal manera que si queremos establecer los permisos de lectura y escritura para un archivo y otorgárselos solo al dueño del archivo, sumaremos:
4 (lectura) + 2 (escritura) = 6 y utilizaremos la orden:

```
chmod 600 <archivo>
```

3.1.9 Interfases de usuario

La interfaz de usuario es la parte del sistema que define cómo interactuamos con él, cómo introduciremos las órdenes y otros tipos de información y cómo nos mostrará el sistema las sugerencias y la información. Incluye tanto la apariencia característica del sistema como su operación. En ocasiones se hace referencia a la interfaz de usuario como el "aspecto y sentir del sistema". Para la mayoría de los usuarios, la interfaz primaria del sistema UNIX ha sido la interfaz de línea de comando suministrada por el shell, la cual ha sido la más importante desde el nacimiento del sistema y ha perdurado a través de los años. Hasta hace poco la línea de comandos era la única interfaz de usuario disponible para UNIX, pero en los últimos años la aparición de interfaces de usuario gráficos (GUI) ha cambiado esto. Las interfaces gráficas de usuario hacen que el trabajo con el sistema UNIX sea más fácil, más efectivo y más divertido.

3.1.9.1 Diferentes Tipos de Shell

Una gran parte del uso del sistema UNIX consiste en dar órdenes al sistema. Cuando se emite una orden se está teniendo relación con el shell, la parte del sistema operativo a través de la cual se controlan los recursos del sistema. El shell proporciona una gran parte de las características que hacen al sistema UNIX ser potente y flexible. El shell es un intérprete de órdenes, un lenguaje de programación y mucho más. Como interprete de órdenes, el shell lee las órdenes introducidas por el usuario y dispone de lo necesario para su ejecución. Además de esto, el shell puede ser utilizado como un lenguaje de programación para crear programas llamados guiones, basados en instrucciones shell. El shell es la interfaz de usuario más sencilla, ya que a través de la línea de comando permite introducir las órdenes al sistema.

Cuando nos conectamos con el sistema, se inicia automáticamente un programa de shell, este es el shell de inicio de sesión. El programa shell particular que se ejecuta cuando accesamos al sistema está determinado por su entrada en el archivo /etc/passwd. Este archivo, como ya lo habíamos mencionado, contiene información que el sistema necesita conocer sobre cada usuario, incluyendo el nombre, el ID de inicio de sesión, etc. El último campo de este archivo contiene el nombre del programa a ejecutar que es el shell.

Existen muchos diferentes tipos de shell y por supuesto cada uno de ellos tiene sus características propias y sus particularidades, aunque la forma de trabajo en general es similar.

A continuación presentamos una tabla con los nombres de los shells existentes en los sistemas UNIX.

Shell	Descripción
Bash	Versión de GNU del shell estándar con características añadidas del shell C, su nombre proviene de la abreviación de "Born again shell".
Csh	El shell C que integra un lenguaje de programación similar al lenguaje C.
Jsh	El shell de trabajo, una extensión del shell estándar.
Ksh	El shell korn
Sh	El shell estándar de UNIX (llamado algunas veces el shell de Bourne).
Tcsh	Una versión mejorada de C shell.
Zsh	Un shell mejorado que se asemeja a ksh

Tabla 3-3 Shells utilizados en UNIX

Dos de los shells más comunes son el korn shell (Bourne shell) y el C shell. Ambos shells tienen la misma finalidad, pero por diferentes caminos. El C shell es realmente una ampliación del korn shell.

El shell C fue desarrollado en la universidad de California de Berkeley como un shell más apto para programadores que el shell Bourne. El shell korn posee todas las características del shell C, pero utiliza la sintaxis del shell Bourne, porque básicamente el shell korn fue creado con el propósito de sustituir al shell Bourne, añadiendo muchas ventajas, con lo cual en muchos sistemas al hacer una llamada al Bourne shell, aparecerá el korn shell y probablemente no nos percataremos de ello. Sin embargo, cualquiera que se ejecute no importa, realmente son muy similares. En sus formas más sencillas, los shells Bourne y Korn utilizan el signo del dólar ($) como indicador estándar; el shell C utiliza el signo de porcentaje (%) como indicador. Afortunadamente o no, estos indicadores pueden cambiarse, por lo que se puede ver o no el signo de dólar o de porcentaje al conectarse al sistema por primera vez.

El shell Bourne, conocido como sh, es el shell original de UNIX. Lo escribió Steve Bourne en colaboración con Jhon Mashey, ambos en los laboratorios Bell de AT&T [Hahn, 1995]. Se encuentra disponible en todos los sistemas UNIX, el programa ejecutable para este shell se encuentra en el archivo /bin/sh. Puesto que el shell Bourne está disponible en todos los sistemas UNIX y posee muchas propiedades que facilitan el trabajo de los usuarios, este shell es uno de los más utilizados.

El shell C, conocido como csh, fue desarrollado por Bill Joy en la Universidad de California Berkeley [Hahn, 1995]. Los estudiantes de Berkeley han ejercido una gran influencia sobre los sistemas UNIX. Dos buenos ejemplos de esta influencia son el shell C y el editor de textos vi. El shell Bourne posee una capacidad superior de programación, pero el shell C fue desarrollado para reflejar el hecho de que la informática se estaba haciendo más interactiva. El programa para ejecutar este shell C se encuentra en el archivo /bin/csh.

La sintaxis del shell C es muy parecida al lenguaje de programación C. Esta es una de las razones por la que los archivos de secuencias de shell escritos para el shell C no pueden ejecutarse a menudo bajo el shell Bourne o korn, existen diferencias. El shell C tiene algunas características útiles que no incluye el shell Bourne, como la edición de órdenes e historial y asignación de alias.

Las capacidades básicas del C shell son:

- Crear notaciones taquigráficas para un comando o una serie de comandos.
- Ejecutar varios trabajos simultáneamente, con o sin nuestra intervención.
- Parar un trabajo y empezar de nuevo.
- Ejecutar comandos usados previamente.
- Personalizar el ambiente a las necesidades personales.
- Escribir programas a nivel de comandos, para realizar cualquier tipo de tarea.

3.1.9.2 Interfaz Gráfica de usuario

Con la aparición del sistema V versión 4 de UNIX, se incorporan interfaces de usuario estándar y sistemas de ventanas para aplicaciones basadas en carácter y gráficas. La interfaz gráfica de usuario incluida con la versión 4 se denomina OPEN LOOK, que ofrece una forma consistente, efectiva y eficaz de interactuar con las aplicaciones. Proporciona varias posibilidades de gestión y operación de ventanas, utilización de barras de desplazamiento, claves de pulsar, pulsadores, menús, ventanas emergentes y facilidades de ayuda. Con esta interfaz los usuarios nuevos pueden aprender rápidamente las aplicaciones. Se proporciona una caja de herramientas gráficas que puede utilizarse para construir aplicaciones OPEN LOOK.

Otra interfaz gráfica de usuario importante es MOTIF, desarrollada originalmente por la Open Software Foundation, ha llegado a ser mucho más popular que OPEN LOOK en los últimos años. MOTIF está incluido en algunas versiones del sistema UNIX. Desde que surgió el Sistema V Versión 4 se ha estado desarrollando una interfaz gráfica de usuario estándar conocida como *Common Desktop Enviroment* (CDE). El CDE ha sido adoptada por muchos de los desarrolladores importantes de software del sistema V.

La versión 4 también proporciona el *Framed Access Command Enviroment* (FACE), que es una interfaz para terminales de usuario basada en caracter. FACE presenta el entorno del sistema UNIX a través de ventanas que contienen menús. Las nuevas herramientas hacen posible la utilización de FACE en las aplicaciones.

El desarrollo y la estandarización de las interfaces gráficas de usuario para UNIX es un área extremadamente activa en la evolución de los sistemas UNIX de sobremesa.

3.1.9.2.1 Gestores de Ventanas en UNIX

El administrador de ventanas en los sistemas X Window del cual hablaremos más delante es una aplicación cliente que suministra la gestión de ventanas básica y las funciones de manipulación que utilizamos al interactuar con el sistema, esto incluye la disposición básica de las ventanas, bordes, apariencia de menús, creación y eliminación de ventanas, movimiento de ventanas, manejo de teclado y trazados de colores, además de la iconización de ventanas. Todas estas funciones hacen que el administrador de ventanas sea el principal

determinante de la apariencia y funcionamiento general de nuestro sistema gráfico en UNIX.

Al igual que el sistema UNIX reforzó la innovación de interfaces de usuario orientadas a caracteres, o "shell", exportando las funciones de interfaz de usuario fuera del sistema operativo a un módulo separado; se permite la existencia de muchas otras versiones como el Shell C, Bourne shell, Korn shell, etc., con lo que el sistema X Window pone su propia interfaz de usuario en un módulo de software separado llamado *Administrador de Ventanas*. Y al igual que la naturaleza modular del shell conduce a shells alternativos también se han desarrollado muchos administradores de ventanas alternativos.

El sistema X Window no dicta un "aspecto y funcionamiento" específico; por ejemplo, una GUI de PC pueden tener dos apariencias y estilos de operación muy diferentes. Podrán diferir en la forma en que los menús y sus acciones son representadas, en la manera en que nuestra aplicación convierte una ventana en un icono y en otras características fundamentales. Esta flexibilidad tiene un valor, las inconsistencias resultantes pueden anular las ventajas potenciales de las interfaces gráficas. Para evitar este problema, se han desarrollado diversos productos que pretenden suministrar una interfaz de usuario consistente tanto para el sistema UNIX completo, como para aplicaciones de diferentes fabricantes. Los administradores de ventanas con mayor amplitud de uso y distribución son el administrador de ventanas *Motif*, *Mwm* de Open Look, *Olwm* de Sun Microsystems, juntos con otros relativos como *Olvwm*, el cual maneja ventana en una "pantalla virtual" mucho mayor que la ventana real que se encuentra delante del usuario.

La mayoría de las aplicaciones del sistema X Window son construidas a partir de objetos de propósitos generales de software reutilizable llamados "widgets y gadtgets". Las bibliotecas de esos objetos son conocidas como toolkits. Los toolkits (kit de herramientas) más conocidos se guían por las convenciones de las interfaces de usuario Motif/CDE u Open Look; Mwm y Olwm fueron escritos para trabajar de una manera consistente con los widgets de los kits de herramientas Motif y Open Look.

Motif tiene un aspecto y presentación que fueron desarrollados por la Open Software Foundation (OSF), basados en el trabajo de DEC y HP. Se diseñó para ser similar al Administrador de Presentaciones de IBM y Microsoft Windows.

Open Look (OL) fue desarrollado por Sun Microsystems y AT&T, basado en el trabajo previo de Xerox y en las GUI Sun previas. Ha sido la GUI del sistema X Window más común sobre plataformas Sun.

Aunque existen diferencias claras en el diseño gráfico y la apariencia entre Motif y Open Look, y aunque existen diferencias en características específicas, tanto Motif como Open Look, parecerán similares a los usuarios de las GUI de PC actuales.

La versión de las herramientas del sistema X Window de Novell "Unixware" nos permite elegir la apariencia de Motif-CDE u OL, dependiendo del valor de una variable de entorno UNIX exportada de nuestro shell.

Debemos tener en mente que estos entornos implícitos compatibles con Motif u Open Look son sólo puntos de comienzo. Conforme desarrollemos hábitos de trabajo y preferencias habituales que optimizan nuestra productividad, estaremos preparados para ajustar nuestro propio entorno de sistema X Window como mejor nos convenga.

3.1.9.2.2 El Sistema X Window

Ahora las interfaces gráficas son de uso común en los PC y las GUI del sistema UNIX comparten con ellas muchas utilidades. No obstante, las interfaces gráficas de usuario para el sistema UNIX tienen algunas características especiales que se ajustan a las necesidades particulares de las aplicaciones bajo sistema UNIX. Los entornos gráficos deben soportar aplicaciones en red, deben permitir que las aplicaciones sean independientes de monitores específicos y de hardware de terminales, deben permitir que las aplicaciones gráficas se puedan mover fácilmente a través de la gran variedad de hardware sobre la que corre el sistema UNIX. El entorno gráfico estándar del sistema UNIX que se ajusta a estas necesidades es el sistema *X Window*.

El sistema X Window es una interfaz gráfica de fácil lectura y un entorno de ventanas para desarrollar y ejecutar aplicaciones con interfaces gráficas de usuario conectadas en red. Es un componente estándar de la mayoría de los sistemas UNIX y también está disponible como un paquete complementario en otros productos software o en una versión de dominio público.

Los principales conceptos en los que se basa el sistema X Window incluyen un modelo cliente-servidor para la interacción entre las aplicaciones y los dispositivos de terminales, un protocolo de red, varias herramientas de software que podrán ser usadas para crear aplicaciones basadas en X Window y una colección de aplicaciones útiles que suministran utilidades básicas para nuestras aplicaciones.

Las Interfaces Gráficas de Usuario (GUI) sustituyen el estilo de interacción con línea de comandos en el sistema UNIX por otro basado en menús, iconos y la selección y manipulación de objetos. En vez de tener que recordar las órdenes y las opciones de dichas órdenes de comando, trabajamos directamente con representaciones gráficas de objetos (archivos, programas, dibujos, listas) y seleccionamos acciones a partir de menús en vez de tener que teclear sus nombres.

El sistema X Window incorpora todas las ventajas de interfaz de usuario de sus contemporáneos (Apple Macintosh, Microsoft Windows, etc.) y a esto le añade algunas otras muy útiles propias del sistema X Window. Al mismo tiempo, continúa la filosofía de UNIX de ser modular y sencillo, y por ello más fácil de invocar, porque los reemplazos experimentales para pequeñas herramientas modulares son más fáciles de construir que los reemplazos para un complejo conglomerado de sistema operativo, sistema de ventanas y un administrador de interfaz de usuario igual al de Apple Macintosh o Microsoft Windows.

El servidor X Window es el software que controla el hardware de la interfaz de usuario, el teclado, el puntero y una o más pantallas que a menudo corren sobre otras plataformas, inclusive sobre computadoras en ejecución bajo sistemas operativos, tales como Microsoft

Windows o Apple Macintosh, e incluso terminales X Window individuales que ejecutan el servidor X desde firmware sin ningún sistema operativo.

El sistema X Window es un sistema de ventanas en red, lo cual quiere decir que aún los servidores X pueden suministrar una interfaz de usuario no sólo a todos aquellos procesos clientes que estén corriendo sobre la misma computadora UNIX o estación de trabajo como el propio servidor X, sino que también a aquellos programas que se estén ejecutando en otras computadoras conectadas a la misma red; incluso puede ser una red muy amplia tal como la red global Internet. Se puede utilizar el servidor X que esté trabajando en una computadora ubicada en la localidad remota desde cualquier otra parte del mundo.

Un servidor es un proceso que permite a su vez a varios otros procesos llamados "clientes", compartir algunos recursos físicos o lógicos. Al igual que un servidor de archivos permite a varios procesos compartir los archivos en un sistema de archivos central, un servidor X permite a varios procesos cliente compartir acceso al hardware que les proporciona una interfaz con el usuario. Este hardware (área de pantalla, teclas del teclado, posición del puntero y botones) necesita ser compartido de alguna manera por los procesos de los clientes, salvo raras excepciones, un servidor X comparte sus recursos entre procesos clientes..

El compartir estos recursos, tales como, el área de la pantalla y las teclas del teclado, se hacen bajo el control del usuario. Este control de usuario requiere una interfaz de usuario interactiva: la mayoría de los administradores de ventanas crean y controlan un macro alrededor de cada ventana cliente para permitir al usuario identificar, mover, reajustar, y calibrar las ventanas de aplicación en la pantalla. La mayoría de los administradores de ventanas también suministran uno o más menús, habitualmente invocados desde el marco de la ventana para funciones específicas de ventanas, o desde el fondo, o ventana raíz, para otras funciones.

El cliente administrador de ventanas es especial en algunos aspectos; por ejemplo, sólo un administrador de ventanas podrá conectarse a un servidor X en un momento dado. Pero en muchos aspectos no es más que otro cliente. Por ejemplo, no se requiere que el administrador de ventanas corra en la misma máquina que el servidor que controla. Para servidores X corriendo sobre hardware dedicado "terminales X" el administrador de ventanas, al igual que todos los clientes, debe ser ejecutado desde sistemas UNIX en red.

3.2 COMANDOS MÁS COMUNES Y EXPRESIONES REGULARES

Durante el transcurso de este capítulo sobre UNIX hemos visto ya algunas instrucciones del sistema, incluyendo la entrada a sesión y algunas otras. Es importante mencionar, aunque ya lo habíamos hecho, que para el sistema UNIX existe una diferencia muy grande entre las letras mayúsculas y las minúsculas, por lo tanto cada instrucción deberá escribirse tal y como se muestre aquí. En UNIX generalmente los comandos se teclean con letras minúsculas.

Ya hemos visto la manera de entrar a un sistema UNIX, para ello debemos contar con una clave y una contraseña que previamente debe ser asignada por el administrador del sistema. Si no es así, es necesario solicitarla a algún proveedor del servicio o institución que pueda otorgárnosla, de otra manera podemos instalar nuestro propio sistema UNIX, ya sea que descarguemos directamente de algún sitio en Internet una versión reciente del sistema Linux, FreeBSD, OpenBSD, etc. que son gratuitos, o podemos también adquirir una versión en CD-ROM por unos pocos pesos, realmente lo que se cobra en estas ediciones es el trabajo de editar el CD y algunas utilerías para facilitar el trabajo de instalación, no precisamente el sistema operativo.

Vamos a comenzar analizando los comandos más comunes que se utilizan en una sesión dentro del sistema operativo UNIX. Iniciaremos trabajando con los comandos para la manipulación de archivos y directorios, tomando en cuenta que dentro de la estructura del sistema de archivos de UNIX existe una jerarquía de directorios en los cuales se organiza la información. Consideremos que es básico saber como manipular dichos archivos y directorios.

3.2.1 Comandos para la manipulación de directorios

El comando cd

cd [opciones] <directorio>

Cambia a un directorio de trabajo. <directorio> representa el nombre y la ubicación o ruta del directorio.

Opciones
.. Regresa al directorio inmediato superior (padre).
~ Indica el directorio base del usuario.
/ Lleva hasta el directorio raíz.
(sin parámetros) Posiciona automáticamente en el directorio base o home.

EJEMPLOS

```
sunserver% cd ~
/home/alumnos/imc

sunserver% cd ..
/home/alumnos.
```

En este ejemplo se hace un cambio al directorio base del usuario y se usa la opción "cd .." para cambiar al directorio inmediato superior (/home/alumnos).

```
sunserver% cd / Cambia al directorio raíz.
```

El comando mkdir

mkdir <directorio(s)>

Crea uno o más directorios. Para hacerlo se debe tener permiso de escritura en el directorio en donde nos encontramos. Generalmente en el directorio base o home.

EJEMPLOS

sunserver% mkdir trabajos

Este ejemplo creará el directorio trabajos dentro del directorio donde se estuviera en ese momento.

sunserver% mkdir /tmp/trabajos

En este ejemplo se crea el directorio trabajos dentro del directorio /tmp sin necesidad de cambiarse a /tmp.

El comando pwd

pwd <Enter>

Permite identificar el directorio en el que se está actualmente.

EJEMPLOS

sunserver% pwd
/home/alumnos/imc
sunserver%

3.2.2 Comandos para la manipulación de archivos

A continuación se describen los principales comandos para manipulación de archivos en el sistema de archivos de UNIX. Los corchetes [] representan el lugar en que se deben ubicar las opciones del comando. Los corchetes angulares <> representan información o nombres de ejemplo; al momento de trabajar deberán ser sustituidos por información real.

El comando awk

awk [-**F**del] ' patrón {acción}
patrón {acción}
.
.
' nombres-e-rchivos
o
awk [-**F**del] –f archivo-patrón-acción nombres-de -archivos

Awk explora cada archivo de entrada buscando registros que satisfagan el patrón, y si hay alguno que coincide, se ejecuta la acción asociada con él. Si las secuencias patrón y acción son largas, es conveniente almacenarlas en un archivo, al que se hace referencia en la sintaxis como "archivo-patrón-acción", y entonces hay que invocar awk con la opción –f tal y como se muestra en la segunda forma. Se puede especificar el delimitadorde campos mediante la opción -F tal como se mostró anteriormente y donde "del" es la cadena que

delimita los campos. Por defecto, los campos para awk están separados por un espacio en blanco.

Los patrones se definen como sigue:

BEGIN
END
/expresión regular/
expresión relacional
patrón && patrón
(patrón)
!patrón
patrón, patrón

Las acciones consisten en una lista de cero o más sentencias separadas por puntos y comas (;) o caracteres de nueva línea. Se pueden emplear corchetes para agrupar sentencias. Awk permite las siguientes sentencias:

if (expresión) sentencia [else sentencia]
while (expresión) sentencia
for (expresión;expresión;expresión) sentencia
break;
continue;
next;
exit;
variable=expresión
print [expresión] > [expresión]
print [expresión] ! [expresión]
for (var in array) sentencia

Comúnmente se usa awk para seleccionar registros basándose en le valor de algún campo. En awk se puede hacer referencia a un campo con el signo de dólar ($) seguido del número del campo. Así, $1 se refiere an campo primero, $2 al segundo, etcétera, $0 referencia el registro completo. Consideremosun pequeño archivo que contiene nombres de estudiantes, especialidad y edad y ejecutemos sobre él un programa ejemplo de awk:

EJEMPLO:

```
$ cat estudiantes
John, P         Física          20
Rick, L         Mecánica        21
Jack, T         electricidad    23
Larry, M        Química         22
Phil, R         Electricidad    21
Mike, T         mecánica        22
Paul, R         Química         23
John, T         Química         23
Tony, N         Química         22
James, R        Electricidad    21
```

```
Awk '$3 > 22 { print $1 }' estudiantes
Jack, T
Paul, R
John, T
```

En este ejemplo, imprimimos el primer campo de todos los registros cuyo tercer campo es mayor que 22. El patrón es "$3 > 22" y la acción consiste en la sentencia: "print $1". Este patrón se puede clasificar como una expresión relacional.
Ahora mostremos los nombres de todos los estudiantes cuya especialidad sea la electricidad. Debido a que al introducir los datos se ha utilizado tanto "Electricidad" como "electricidad", tenemos que usar expresiones regulares para realizar la búsqueda.

EJEMPLO:

```
$ awk '$2 ~ /[Ee]lectricidad/ { print $1} ' estudiantes
Jack, T
Phil, R
James, R
```

El operador relacional ~ se emplea pra comprarcon expresiones regulares. !~ significa que se satisface la condición si no hay coincidencia con la expresión regular.
He aquí algunos otros ejemplos de utilización de awk:

```
$ awk '$0 ~ /^J.*/ {print $0} ' estudiantes
John, P         Física          20
Jack, T         electricidad    23
John, T Química         23
James, R        Electricidad    21

$ awk '$1 ¡~ /[A-Zb-z] *a [A-Zb-z] */ {print $0} ' estudiantes
John, P         Física          20
Rick, L Mecánica        21
Phil, R         Electricidad    21
Mike, T         mecánica        22
John, T Química         23
Tony, NQuímica          22
```

En el primer ejemplo, mostramos todos los campos de todos los registros que comienzan con la letra J. En el segundo ejemplo, mostramos todos los campos de todos los registros cuyo primer campo no contiene ninguna a minúscula.

El comando cat

cat [opciones] <nombre de archivo> Con este comando se puede desplegar el contenido de un archivo de texto.

Opciones

-v hace visibles los caracteres no imprimibles.
-e Indica el fin de una línea con un signo de pesos.
-t muestra tabuladores.

EJEMPLOS

```
sunserver% cat /usr/README
Netscape 5.0 (X11) IMPORTANT! Before going any further, please read and accept the terms in the
file LICENCE.
```

En el ejemplo anterior se observa cómo cat despliega un archivo de texto utilizando la ruta completa para llegar a él.

El comando cmp

cmp <archivo1> <archivo2>

Compara byte por byte de <archivo1> con <archivo2>. Usa la entrada estándar. Si los archivos son diferentes reporta los bytes y el número de línea dónde encuentra la diferencia.

EJEMPLOS

```
sunserver% cat archivo1
Perro
Gato
Conejo
Ave
sunserver% cat archivo2
Perro
Canario
Gato
Conejo
Ave
sunserver% cmp archivo1 archivo2
archivo1 archivo2 differ: char 7, line 2
```

El comando cp

cp <archivo fuente> <archivo destino>
cp <lista de archivos> <directorio destino>
Copia <archivo fuente> en <archivo destino>, o copia o más archivos (<lista de archivos>) con el mismo nombre bajo un directorio (<directorio destino>).
Si <archivo destino> es un archivo existente éste se sobre escribe; si <archivo destino> es un directorio, el archivo es copiado dentro del directorio.

EJEMPLOS

```
sunserver% cp carta carta.sav
```

En este ejemplo se copió el archivo carta en carta.sav, quedando la copia en el directorio actual.

```
sunserver% cp comandos /tmp
```

Copia el archivo comandos, con el mismo nombre en el directorio /tmp.

El comando cut

cut [opciones] <archivo(s)>

Remueve columnas o campos seleccionados en uno o más archivos.

Opciones
-clista Corta los caracteres en las posiciones especificadas en la lista (lista es una secuencia de enteros. Se debe usar una coma para separar valores y un guión para especificar rangos.

-dc Especifica el delimitador de campos (c), por omisión éste es espacio o TAB. Regularmente se usa con la opción –f.

-flista Por posición de campo. lista contiene enteros que son el número de campo a ser cortados.

EJEMPLOS

```
sunserver% cut –d: -f1 /etc/passwd
aavina
avaidovi
adehmlow
cblanco
alaveaga
alchave
agodfrey
alliance
alramire
```

En el ejemplo anterior se indicó que: cortará con el delimitador ":" (-d:) el campo número uno (-f1) del archivo /etc/passwd

El comando diff

diff [opciones] <archivo1> <archivo2> Despliega línea por línea las diferencias entre dos archivos.

Opciones
-w Ignora espacios y TAB
-r Aplica diff recursivamente.

El comando ed

ed <archivo>

Editor de texto estándar. Si <archivo> no existe ed lo crea; en otro caso ed lo abre para editarlo.

El comando find

find path [criterios] <lista de directorios> <expresión> [criterios]

Busca de manera descendente, recursiva y jerárquica por cada ruta en la lista de rutas (pueden ser una o más) para encontrar archivos que concuerden con <expresión> especificada en el comando; find no los mostrará en pantalla, a menos que se de una orden específica para hacerlo.

Criterios

-name <archivo>	El archivo avaluado satisface esta condición si <archivo> equivale a su nombre.
-user <usuario>	El archivo evaluado satisface esta condición si pertenece al usuario especificado en <usuario>.
-print	El nombre del archivo se despliega.

EJEMPLOS

```
sunserver% find . –name "c*" –print
./.desktop-poe/configchecks
./.desktop-poe/configchecks/checkversion
./.netscape-cache/cache2FE1F9A2000570D
./.netscape-cache/cache2FE1F9A5001570D.gif
./.netscape-cache/cache2FE1F9A5002570D.gif
./.netscape-cache/cache2FE1F9A5003570D.gif
./.netscape-cache/cache2FE1F9A5004570D.gif
./.netscape-cache/cache2FE1F9A5005570D.gif
./.netscape-cache/cache2FE1F9A5006570D.gif
```

La línea de comando anterior localiza los nombres de todos los archivos que se encuentran en el directorio de trabajo y sus subdirectorios que comienzan con la letra "c".

El comando grep

grep [opciones] <expresión> <archivo(s)>

Busca en <archivo(s)> las lineas que contienen <expresión>.

Opciones

-c Imprime sólo el número de líneas en que aparece <expresión>.
-i No hace distinción entre mayúsculas y minúsculas.
-n Imprime las líneas y sus números.
-v Imprime todas las líneas que no contienen <expresión>.

El comando head

head [opciones] <archivo>

El comando head despliega las primeras líneas de un archivo (el predeterminado es diez).

Opciones
-n Donde *n* es el número de líneas a ser desplegadas.

EJEMPLOS
```
sunserver% head –5 frutas
naranja
pera
manzana
melón
sandía
```

El comando ln

ln [opciones] <archivo existente> <liga>
ln [opciones] <archivo existente> <directorio>

Crea seudónimos (ligas) a un archivo, permitiendo el acceso por diferentes nombres.
En el primer caso <archivo existente> a <liga>, donde <liga> es un archivo nuevo. En el segundo caso las ligas se crean con el mismo nombre en el directorio <directorio>.

Opciones
-f Forza la creación de la liga.
-s Crea una liga simbólica.

EJEMPLOS
```
sunserver% ln /usr/alex/literatura/memo2
```

El ejemplo anterior crea un enlace entre el directorio de trabajo y el archivo memo2 del directorio /usr/alex/literatura. El archivo aparece en el directorio de trabajo como memo2.

El comando ls

ls [-lafg] <directorio>

Lista el contenido de directorios. Si no se proporciona <directorio>, lista los archivos del directorio actual.

Opciones
-a (Todos) Esta opción provoca que ls despliegue información sobre todos los archivos, incluidos los ocultos, en el directorio actual.
-g (grupo) Despliega el identificador del grupo. Se usa junto con la opción –l, entonces se reemplaza el nombre de usuario que esté desplegando con el nombre del grupo.
-l (formato largo) Despliega siete columnas de información sobre cada archivo. Estas columnas se describen a continuación:

- La primera columna contiene 11 caracteres que hacen referencia al tipo de archivo. El primero puede tener los valores que se listan a continuación, y los siguientes caracteres representan los permisos de acceso para el dueño del archivo, el grupo al que pertenece y todas las demás personas.

  Tipos de archivo:
  - Archivo ordinario
  b Archivo bloque
  c Archivo caracter
  d Directorio
  l Liga

- La segunda columna informa el número de enlaces que tiene el archivo.
- La tercera despliega el nombre del dueño del archivo.
- La cuarta columna indica el tamaño del archivo en bytes o los números mayor y menor si se trata de un archivo especial. En el caso de un directorio, esta columna representa el tamaño del directorio.
- Las siguientes dos columnas despliegan la fecha y la hora de la última modificación hecha al archivo.
- La última columna es la del nombre del archivo.

-r (invertir) Despliega una lista de los archivos en orden alfabético inverso.
-F Coloca un asterisco (*) junto a los archivos ejecutables, marca una arroba (@) para indicar ligas y una diagonal (/) para indicar subdirectorios.

EJEMPLOS

```
sunserver% ls –l
total 43 drwxr-xr-x 2 adag 512 Feb 6 15:07 REPORTES
drwxr-xr-x      2 adag 512 Nov 10  14:52 WEB
-rw-r—r--       3 adag 237 Feb 8   16:00 comandos
```

En el ejemplo anterior se teclea el comando *ls* junto con el parámetro *–l*, lo que nos proporciona el listado en formato largo del directorio actual. Las columnas presentadas contienen la información ya explicada.

El comando more

more <archivo>

Despliega <archivo> en una terminal, una pantalla por vez; es similar a cat, pero se hace una pausa cada vez que se muestra una pantalla. Como respuesta al comando more pueden utilizarse:

<Barra espaciadora> para ver la siguiente página del archivo.
<Enter> Para ver otra línea.
q Para terminar.

EJEMPLOS

```
sunserver% more archie.dir
Austria         archie.univie.ac.at         131.130.1.23
Belgium         archie.belnet.be            193.190.248.18
Canada          archie.bunyip.com           192.77.55.2
Canada          archie.uqam.ca              132.208.250.10
Finland         archie.funet.fi             128.214.6.102
France          archie.univ-rennes1.fr      129.20.254.2
Germany         archie.th-darmstadt.de      130.83.22.1
Israel          archie.ac.il                132.65.16.8
Italy           archie.unipi.it             131.114.21.10
Japan           archie.wide.ad.jp           133.4.3.6
Korea           archie.hana.nm.kr           128.134.1.1
Korea           archie.kornet.nm.kr         168.126.63.10
Korea           archie.sogang.ac.kr         163.239.1.11
Norway          archie.uninett.no           128.39.2.20
Poland          archie.icm.edu.pl           148.81.209.2
Spain           archie.rediris.es           130.206.1.2
Sweden          archie.luth.se              130.240.12.23
Taiwan          archie.ncu.adu.tw           129.83.166.12
- More – (58 %)
```

En el ejemplo, el comando more sólo desplegó el 58 % del total del documento en una pantalla. Para desplegar otra pantalla, es necesario presionar la barra espaciadora.

El comando mv

mv <archivo1> <archivo2>

Mueve o renombra archivos o directorios. Trabaja de la siguiente manera:

archivo1	archivo 2	Resultado
Archivo	nombre	Renombra archivo como nombre.
Archivo	archivo	Sobrescribe archivo en archivo existente
Directorio	nombre	Renombra directorio como nombre
Directorio	Directorio existente	Mueve directorio como un subdirectorio de directorio existente.
Archivo	Directorio	Mueve archivo dentro de directorio

Tabla 3-4 Esquema de trabajo del comando mv

EJEMPLOS

```
sunserver% mv carta carta.1201
```

Esta línea de comandos renombra el archivo carta del directorio de trabajo a carta.1201.

```
sunserver% mv carta.1201 /usr/archivos
```

El ejemplo anterior mueve el archivo de lugar para que aparezca con el mismo nombre dentro del directorio /usr/archivos.

El comando rm

rm [opciones] <archivo(s)>

Borra uno o más archivos. Para hacerlo, se debe tener derecho de escritura sobre el directorio que contiene <archivo(s)>.

Opciones
-r (recursivo) Si <archivo> es un directorio, remueve también su contenido incluyendo subdirectorios.
-i (interactivo) Pregunta antes de borrar cada archivo.

EJEMPLOS
sunserver% rm memo
Borra el archivo memo que se encuentran en el directorio actual.

El comando sort

sort [opciones] <archivo(s)>
Ordena las líneas de <archivo(s)>.

Opciones
-b Ignora espacios iniciales en blanco.
-c verifica si <archivo(s)> ya está(n) ordenado(s). En caso de que así sea no produce una salida.
-d Ordena en forma de diccionario. Con esta opción sort ignora todos los caracteres que no son alfanuméricos o blancos. En especial no considera los caracteres y signos de puntuación ni los de control.
-f Ignora diferencias entre mayúsculas y minúsculas.
-n En orden aritmético.
-u Imprime sólo una vez las líneas iguales.

EJEMPLOS
sunserver% cat lista

Torobio Prado	94201
Juan Dávila	94111
Alicia MacLeod	94114
David Mack	94114
Toño Bermejo	95020
Julio Campos	84072
Ricardo MacDonald	95510
Ada Cabrales	95224
Lorena Ramos	96222
León Rodríguez	94666
Edmundo Vicario	96999

```
sunserver% sort lista
Ada Cabrales        95224
Alicia MacLeod      94114
David Mack          94114
Edmundo Vicario     96999
Juan Dávila         94111
Juan Campos         84072
Lorena Ramos        96222
León Rodríguez      94666
Ricardo McDonald    95510
Toño Bermejo        95020
Toribio Prado       94201
```

El comando tail

tail [opciones] <archivo>
Imprime las últimas líneas de <archivo>, por lo regular las últimas 10.

Opciones
-n Donde n representa el número de líneas que se desea ver.

EJEMPLOS

```
sunserver% tail –2 cuestionario
```

Que lo cambie y que le explique donde se guarda toda la información con respecto a una cuenta de correo electrónico.

El comando vi

vi [opciones] <archivos>
Editor de textos visual de pantalla completa. Lo analizaremos más adelante.

El comando chown

chown <nuevo dueño> <archivo(s)>
Cambia la propiedad de un archivo a <nuevo dueño>. <nuevo dueño> es un número de usuario (UID) o un nombre de usuario habilitado en el archivo /etc/passwd. Solo el actual dueño del archivo o el superusuario pueden realizarlo.

EJEMPLOS

```
sunserver% ls –l comandos
-rw-r—r— 1 adag 273 Feb 8 16:00 comandos
sunserver% chown lorena comandos
-rw-r—r— 1 lorena 273 Feb 8 16:00 comandos
```

En el ejemplo anterior fue posible darse cuenta por medio de ls-l que existía un archivo llamado comandos, el cual pertenecía al usuario adag. Mediante el comando chown se designó como nuevo dueño de comandos al usuario lorena.

El comando chgrp

chgrp <grupo> <archivo(s)>
Cambia el grupo de uno a más archivos a <grupo>, <grupo> es el número (GID) o nombre de un grupo que se encuentra en el archivo /etc/group. Sólo el dueño del archivo o el superusuario pueden ejecutar este comando.

EJEMPLOS

```
sunserver% ls –lg comandos
-rw-r—r— 1 adag guests 273 Feb 8 comandos
sunserver% chgrp alumnos comandos
sunserver% ls –lg comandos
-rw-r—r— 1 adag alumnos 273 Feb 8 comandos
```

En este ejemplo se cambia el archivo comandos de su grupo original (guests) al grupo alumnos.

El comando chmod

chmod <permisos> <permisos>
Cambia el modo de acceso a uno o más archivos. Sólo el dueño y el superusuario pueden cambiar los permisos a un archivo. Para saber qué permisos de acceso tiene <archivo>, basta ejecutar el comando ls –l <archivo>. La primera columna o el primer campo de la salida de este comando nos proporciona la información que necesitamos. Como ya se dijo, el primer caracter de los diez que lo forman indica el tipo de archivo; los nueve caracteres siguientes se dividen en tres grupos de tres caracteres y representan los permisos para los usuarios

3.2.3 Comandos Para Trabajo en Red

Los programas de utilerías para red en UNIX son de gran importancia en el sistema, porque permiten a los usuarios realizar operaciones a través de la red. A continuación daremos una descripción de las utilerías necesarias para saber quién ha entrado al sistema y comunicarse con otros usuarios que se sirven de la computadora.

El comando finger

El comando finger despliega información acerca de los usuarios que están trabajando en un sistema UNIX. La sintaxis de este comando es:

finger <@nombre_del_sistema>
o
finger <identificador_de_usuario>

Por omisión el comando finger despliega información acerca de los usuarios que se encuentran conectados al mismo segmento de red, máquina o computadora que nosotros. La información que se proporciona de cada usuario es la siguiente:

- Identificador.
- Nombre completo.
- Nombre de la terminal.
- Tiempo de inactividad (tiempo que lleva el usuario dentro del sistema pero sin trabajar - *idle time*).
- Localización y nombre de las máquinas desde las cuales se conectan los usuarios (si es que el sistema las reconoce).

El tiempo de inactividad se mide en minutos enteros, es decir, no muestra fracciones menores a un minuto. Si aparece el caracter (:) Se mide en horas y minutos; también se puede medir en días y horas, cuando el caracter *d* está presente.

Cuando se da uno o más argumentos a la hora de ejecutar el comando, se despliega información más detallada sobre el usuario que indicó como argumento, esto ocurre aún si la persona no está conectada al sistema en ese momento. La información que se proporciona como argumento puede ser el apellido o el nombre del usuario. La información solicitada sobre el usuario aparece en un formato multilínea que incluye información como la siguiente:

- La localización del directorio de trabajo del usuario y el shell bajo el cual se encuentra el usuario.
- El tiempo que lleva el usuario conectado al sistema, si está trabajando aún; en caso contrario despliega la fecha y la hora de la última vez que el usuario entró al sistema y la máquina desde la cual estableció la conexión.
- La última fecha en que dicho usuario recibió y leyó su correo electrónico. (Esta opción puede ser inhibida).
- Despliega al plan del usuario. Es decir, el contenido del archivo *.plan* que se encuentra en el directorio de trabajo del usuario.
- Los proyectos que el usuario tiene pendientes. Esta información se encuentra graduada en el archivo .proyect de cada usuario.

Si el nombre que se da como argumento al comando finger comienza con @ entonces dicho nombre no se tomará como el identificador de nombre de un usuario, sino como el nombre de alguna máquina y finger desplegará una lista con los usuarios conectados a dicha máquina.

Opciones

-l Provoca que la información aparezca en formato largo.

EJEMPLOS

```
sunserver% finger @ghost
[ghost]
```

Login	Name	TTY	Idle	When	Where
mmorales	Martín Morales	p5	29	Sat 10:39	148.220.50.52
cuentas	Cuentas de serv.	p0	1:58	Sat 10:05	148.220.1.5
carlosz	Carlos H. Zarina	p1	1d	Fri 11:57	148.220.8.75

En el ejemplo anterior, como una arroba (@) precede al parámetro proporcionado, el comando finger toma ese parámetro como nombre de una máquina (ghost) y despliega una lista con los usuarios conectados a ella.

```
sunserver% finger leonf
Login name: leonf                    In real life: Leon Felipe Rodriguez
Directory:/home/alumnos/leonf  Shell: /bin/csh
Last Login Fri feb 16 10:33 on ttyp4 from 148.220.8.91
New mail received Fri Feb 16 09:19:08 1999
Unread since Thu Feb 15 19:12:35 1999
No Plan.
```

En este otro ejemplo se desplegó la información del usuario leonf.

El comando who

El comando who despliega información acerca de los usuarios que están trabajando en un sistema. Este comando tiene la siguiente sintaxis:

who [opciones]

Opciones
Sin opciones Muestra una lista de los usuarios que están trabajando en el sistema, desplegando:

- Identificador.
- Nombre de la terminal.
- Tiempo que lleva cada usuario conectado al sistema.
- Nombre de la máquina desde donde se hace la conexión.

Am i Esta opción despliega lo siguiente:

- Nombre de la máquina.
- Identificador del usuario con el que se está trabajando.
- Nombre de la terminal.
- Tiempo que lleva el usuario conectado al sistema.
- Desde dónde se conecta.

EJEMPLOS

```
sunserver% who

mmorfin      ttyp2   Feb 17 10:00   (148.220.1.5)
rll81575     ttyp6   Feb 17 09:56   (anubis)
curso3       ttypc   Feb 17 12:44   (148.220.1.8)
```

El comando who despliega una lista con los usuarios conectados a la máquina actual (sunserver).

```
sunserver% who am i
imc      pts/4      Nov  9 15:33   (148.220.8.79)
```

Suele ocurrir que al llegar al laboratorio de cómputo, una de las computadoras está encendida con una sesión abierta y no se encuentra el usuario que la estaba utilizando, entonces se utiliza el comando *who am i* para saber quien estaba trabajando en ella y/o si la máquina está disponible.

El comando w

El comando w además de desplegar información acerca de los usuarios que están trabajando en un sistema, muestra lo que cada uno de ellos está haciendo. Su sintaxis es la siguiente:

w <nombre_de_usuario>

EJEMPLOS

```
sunserver% w
1:00pm up 10 days, 3:55, 3 users, load average: 0.00, 0.00, 0.00
User     tty      login@    idle   JCPU   PCPU   what
mmorfin  ttyp2    10:00am   3:03                 -csh
cga86055 ttypd    11:41am   20     1             finger
curso1   ttyp1    1:06pm           1             telnet
```

En el ejemplo anterior la línea de encabezado muestra la hora actual, desde cuándo el sistema está funcionando, el número de usuarios que se encuentran conectados, el promedio de trabajos que ha realizado la máquina hace 1, 5 o 15 minutos. Los campos de información desplegados son: el identificador de cada uno de los usuarios, el nombre del tipo de terminal o tty que está utilizando, la hora y fecha en que cada usuario se conectó al sistema, el tiempo que el usuario lleva sin teclear nada, el tiempo de CPU utilizado por todos los procesos (padres e hijos) en una terminal, el tiempo de CPU que está utilizando el proceso activo y los nombres de los procesos actuales.

El comando users

El comando users despliega una lista de los identificadores de todos los usuarios que se encuentran conectados al sistema en ese momento. La sintaxis del comando es:

users <enter>

EJEMPLOS

```
sunserver% users
imc adag leonf horacio
```

Por medio del comando users, es posible darse cuenta de que la máquina sunserver se encuentran trabajando los usuarios imc, adag, leonf y horacio.

El comando rusers

Este comando permite saber quién está dentro de toda la red. La sintaxis del comando rusers es la siguiente:

rusers <máquina>

La salida que produce rusers es similar a la de los comandos users y who, con la diferencia de que rusers se utiliza únicamente para máquinas remotas. Se hace una llamada de transmisión a la red y se presenta la respuesta de cada una de las máquinas. Normalmente la información se despliega conforme se va recibiendo, pero el orden puede cambiar de acuerdo con los argumentos que se incluyan.

Por omisión se imprime una lista parecida a la que produce el comando users, pero con una línea por máquina.

EJEMPLOS

```
sunserver% rusers
Cuevas hugor
carloz
madrid vg03323 bbsadm
```

El comando write

El comando write se emplea para enviar mensajes a otro usuario que se encuentre dentro del sistema. Cuando se ejecuta el comando write, éste despliega un aviso en la terminal del otro usuario indicándole que está a punto de recibir un mensaje. A continuación se muestra la sintaxis del comando:

write <nombre_de_usuario>
mensaje

<nombre_de_usuario> es el identificador de la persona con quien se desea comunicarse. write copia texto de una terminal en la otra, línea por línea. Cuando se quiere terminar la comunicación se oprimen las teclas Control-D, lo cual indica a write que se detenga, despliegue EOF (fin de archivo) en la terminal del otro usuario y devuelva el primero al shell.

EJEMPLOS

```
sunserver% write lorena
Hola ¡Lorena! Cómo estas??? Te veo a las 5 en el café de aquí enfrente :)
Ada
sunserver%
```

Así es como se vería la terminal de quien envía el write al usuario lorena. A continuación se muestra qué aparecería en la terminal del usuario lorena.

```
sunserver%
Message from adag@sunserver on ttypd at 13:57...
Hola ¡Lorena! Cómo estas??? Te veo a las 5 en el café de aquí enfrente :)
Ada
EOF
sunserver%
```

El comando talk

La orden write copia lo que tecleamos en la terminal y lo visualiza sobre la pantalla de la terminal del otro usuario, sin embargo los caracteres escritos no se visualizan en la otra terminal hasta que se presiona la tecla <Enter>. El comando talk es un programa desarrollado con el propósito de permitir al usuario entablar una conversación en modo carácter desde una terminal a otra. talk anuncia al otro usuario que es solicitado para entablar una conversación. Por ejemplo, si el usuario imc teclea:

```
sunserver% talk adri
```

El comando talk notifica al usuario adri que el usuario imc desea entablar una conversación y solicita la aceptación. El usuario adri ve lo siguiente en su pantalla:

```
Message from Talk_Daemon@sunserver at 18:50 ...
talk: connection requested by imc@sunserver.
talk: respond with:  talk imc@sunserver.
```

Si adri responde con la instrucción: *talk imc@sunserver*, talk divide la pantalla de cada terminal en una mitad superior y otra inferior. Las líneas que se teclean en nuestra terminal aparecen en la mitad superior, y las que teclea el otro usuario aparecen en la mitad inferior. Desde las dos terminales se puede teclear al mismo tiempo y ver cada uno la salida del otro sobre su pantalla. talk permite visualizar carácter por carácter cuando este es tecleado en tiempo real.

El comando mesg

Los comandos write y mesg permiten a un usuario teclar un mensaje que se visualizara en la terminal de otro usuario. Algunas veces, esto puede resultar desconcertante para un usuario que está concentrado en su trabajo. Para ello, UNIX proporciona el comando mesg, que permite aceptar o rechazar mensajes provenientes de otros usuarios con las órdenes talk o write.

```
sunserver% mesg n
```

Desactiva el modo de recepción de mensajes provenientes de otra terminal, el que emite el mensaje verá lo siguiente sobre su pantalla:

```
Permission denied
```

Con la orden:

```
sunserver% mesg y
```

Se restaura el permiso para recepción de mensajes, se puede saber si un usuario ha denegado el permiso de escribir sobre su terminal. Esto es posible utilizando el comando finger y verificando que si el usuario está marcado por un símbolo de asterisco (*), si es así, entonces ha denegado el permiso y será imposible enviarle un mensaje.

El comando traceroute

Este comando visualiza la ruta que los paquetes recorren cuando estos son enviados a través de la red. Puede ser de utilidad para determinar que puntos de la red están creando retardos. La sintaxis se muestra a continuación:

traceroute <nombre_del_servidor>

Esto visualizará el nombre y número de hosts entre la máquina que ejecuto el comando y el servidor elegido.

El comando nslookup

El comando nslookup muestra el IP de un nombre de dominio o viceversa, muestra el nombre de dominio de una dirección IP. La sintaxis es la siguiente:

nslookup <dominio / IP>

El comando ping

Es utilizado para enviar un paquete a un servidor y obtener una respuesta, puede ser utilizado para saber si un servidor está activo (trabajando) o para medir el tiempo de respuesta del mismo. La sintaxis es:

ping <servidor>

El comando whois

whois contacta la base de datos de Internic para la consulta de un dominio, retornando los datos de ese dominio. Esta es la manera de verificar la disponibilidad de un dominio. La sintaxis es la siguiente:

whois <nombre_de_dominio>

3.2.4 Comandos de administración

UNIX es un sistema operativo cuya versatilidad lo ha llevado a ser integrador en múltiples ambientes, generalmente de redes, dadas sus características de multiusuario y multitarea. La administración de UNIX dentro de una red añade a las tareas del administrador el configurar y mantener servicios y recursos que sean compartidos y que respondan a las necesidades de los usuarios de la red.

3.2.4.1 Utilerías del administrador

El comando cron

En algunas ocasiones es útil poder tener un script que sea ejecutado sin la intervención del administrador. En los sistemas UNIX la ejecución de estos comandos de manera periódica sin la intervención del administrador se lleva a cabo por el comando cron, el cual se ejecuta al momento de inicializar la máquina y se mantiene en ejecución mientras el sistema se encuentre operando. Para su ejecución, cron lee uno o más archivos de configuración que contienen una lista de comandos, tiempos en que deberán ser ejecutados y (en algunos sistemas) el nombre de usuario bajo el cual se va a ejecutar el comando.

El archivo de configuración del cron se llama **crontab**. Los sistemas basados en BSD mantienen ese archivo en /etc/crontab o /usr/lib/crontab, los cuales pueden ser modificados únicamente por el superusuario, y en ellos se puede incluir información referente al usuario que ejecutará el comando.

Los sistemas basados en System V tienen un modelo más flexible; en lugar de tener un solo archivo de cron, contienen un directorio llamado crontab [Nemeth, 1995]. Un comando llamado crontab es utilizado para direccionar archivos de crontab al directorio antes mencionado, y el cron lee el contenido de cada uno de los archivos presentes. Generalmente hay un archivo de crontab por cada usuario que desee usar esta utilería. El nombre del archivo es el mismo que el nombre de la cuenta del usuario que desea utilizar el cron.

Dada la flexibilidad que presenta este modelo de manejo de cron, algunos sistemas operativos derivados de BSD han adoptado esta funcionalidad, como es el caso de SunOS.

Normalmente, cron se ejecuta en modo silencioso, pero algunas versiones permiten mantener un archivo bitácora que registra la actividad de los comandos que fueron ejecutados. En los sistemas operativos como Solaris, SunOS e IRIX la bitácora será generada si se activa esta opción en el archivo /etc/default/cron, poniendo la opción YES en la variable CRONLOG.

```
CRONLOG=YES
```

El formato de un archivo crontab en system V es:

Minute Hour Day Mount Weekday Command

para BSD

Minute Hour Day Mount Weekday User Command

Las interpretaciones de cada una de las variables de tiempo se interpretan de la siguiente forma:

Campo	Descripción	Rango
Minute	Minuto dentro de una hora	0 a 59
Hour	Hora del día	0 a 23
Day	Día del mes	1 a 31
mount	Mes del año	1 a 12
weekday	Día de la semana	1 a 7 (BSD) 1=Lunes 0 a 6 (SV) = Domingo

Tabla 3-5 variables de tiempo en el archivo crontab

Cada uno de los campos de tiempo puede contener:

- Un asterisco, el cual indica que cualquier valor es válido.
- Un número entero, el cual indica el valor exacto en el campo de tiempo correspondiente.
- Enteros separados por comas, lo cual indica que cualquiera de esos valores es válido.
- Dos enteros separados por un guión, indicando un rango de valores válido.

Por ejemplo, para especificar la ejecución de un comando los días lunes a viernes a las 10:45 am:

```
45 10 * * 1-5 command
```

Algunos comandos válidos para crontab son:

```
0, 15, 30, 45, 78-18 * * * /usr/lib/atrun
0 0 * * * find / -name ".bak" –type f -atime +7 –exec rm{} \;
0 4 * * * /bin/sh /var/adm/mon_disk 2>&1 >/var/adm/disklog
0 2 * * * /bin/sh /usr/local/ sbin/sec_check 2>&1 | mail root
30 11 31 12 * /etc/wall%Feliz año nuevo!%Los mejores deseos a todos
```

Si se utiliza el signo de porcentaje (%) dentro del comando, le estará indicando a cron el texto que sigue después del signo es una entrada estándar del comando. Cualquier otro caracter de porcentaje en la misma línea indicará que el texto será dividido por líneas.

El comando crontab

En los sistemas BSD se necesita ser el superusuario para poder modificar al archivo de crontab, y una vez modificados se le tiene que enviar una señal de "hangup" al cron para que se reinicialice y tome los nuevos cambios.

En el System V, se utiliza el comando crontab para modificar y crear archivos que serán utilizados por crontab. Las opciones que se pueden emplear son:

Opción	Significado
crontab –e	Edita la tabla de crontab del usuario
crontab –l	Lista el contenido de la tabla de crontab en actual ejecución
crontab –r	Elimina el archivo de crontab del usuario
crontab filename	

Tabla 3-6 Opciones para el comando crontab

En Sistema V el edministrador de la máquina puede especificar quién puede utilizar las ventajas de cron. Esto se logra especificando una línea por usuario si se le va a permitir utilizarlo. Estos archivos de configuración son llamados **cron.allow** y **cron.deny**.

El comando cron verificará primero en el archivo cron.allow quiénes pueden invocar de manera satisfactoria el comando crontab. Si este archivo no existe, entonces verifica el contenido del archivo cron.deny para determinar cuales usuarios no pueden utilizar el crontab.

El comando at

El comando at ejecuta un script (o programa con una serie de comandos) a un tiempo específico.

at [-cm] time [date] [+increment][script]

Opciones
-c Utiliza chs para ejecutar el script. Por omisión se ejecutan en Bourne shell (sh).
-m Envía mensaje de correo al usuario una vez que termina la ejecución.

Existe un archivo at.deny y uno at.allow que controla a los usuarios. Es similar a los ya definidos en crontab.

EJEMPLOS
Ejecutar el comando at con un tiempo específico:

```
# at 1900
> cd /
> ^D <EOT>
Job 7 at Tue Feb 19 19:00:00 1999
```

El comando syslog

Syslog es una herramienta para guardar en bitácoras la actividad generada por el kernel y las utilerías del sistema. tiene dos funciones importantes:

- Libera los programas de la tarea de mantener los archivos de almacenamiento de la información que ellos generan.
- Ayuda al administrador a mantener el control de la actividad del sistema.

Antes cada programa mantenía sus propias bitácoras y el administrador no tenía control de la información que éstos generaban.

Su diseño proporciona una gran flexibilidad, y permite clasificar la importancia de los mensajes y direccionarlos a una variedad de destinos: archivos de bitácora, terminales de los usuarios o hasta otras máquinas.

Syslog está compuesto de varias partes:

syslogd y */etc/syslog.conf*, que son el daemon que realiza la función de rastreo y su archivo de configuración.

openlog, syslog y icloselog son rutinas de librerías que utilizan los programadores para enviar la información a syslog. logger es un comando de usuario para enviar peticiones de rastreo a syslog.

La localización de syslogd se encuentra generalmente en */etc*, */usr/etc* o */usr/sbin*. El programa logger en */usr/ucb*. Las rutinas de las librerías son parte del estándar de librerías de C.

Syslog se inicializa al momento del arranque del sistema y continua su ejecución hasta que el sistema se da de baja o alguien lo elimina de manera manual. El número de proceso en ejecución se almacena en /etc/syslog.pid (en otros sistemas se encuentra bajo /var/run/syslog.pid).

El archivo de configuración en /etc/syslog.conf controla el comportamiento de syslog. En un archivo tipo texto con un formato relativamente sencillo. Las líneas en blanco o que inician con el signo # se toman como comentarios. El formato básico es:

selector <TAB> **action**

Los campos selector y action deben estar separados por uno o más tabuladores; los espacios en blanco no son válidos y traen como resultado la creación de errores difíciles de encontrar.

El campo selector identifica el programa "facility" que está enviando la información a ser almacenada así como la importancia del mensaje bajo la siguiente sintaxis:

facility.level

Tanto los nombres de las facilidades como su nivel de importancia pueden ser elegidos de una pequeña lista de valores definidos, ya que los programas no pueden tener sus propios valores arbitrarios. Existen facilidades definidas para el kernel, para grupos de comandos o utilerías, y para programas desarrollados para su integración al sistema. Los selectors pueden tener las definiciones de "*" y "none" que significan todas las opciones y ninguna respectivamente. Algunas opciones válidas para formar y combinar selectors pueden ser:

Facility.level	action
Facility1, facility2.level	action
Facility1.level; facility2.level2	action
*.Level	action
*.Level; badfacility.none	action

Los valores permitidos para syslog dentro de las facility son:

Facilidad	Programas que la utilizan
kern	El kernel
user	Procesos del usuario
mail	El sistema de correo
auth(security)	Comandos relacionados con seguridad
lpr	El sistema de impresión de BSD
news	El sistema de noticias USENET
uucp	Reservado para UUCP
cron	El daemon de cron
mark	Marcas de tiempo en intervalos regulares
local0-7	Utilizados para mensajes locales
syslog1	Mensajes internos del syslog
authpriv1	Mensajes privados de autorización
ftp	El daemon de FTP

Tabla 3-7 valores permitidos para syslog

Los niveles de importancia se describen en la siguiente tabla de acuerdo con su prioridad. Internamente son representados por pequeños enteros: 0 representa emerg, 1 alert y así consecutivamente.

Nivel	Significado apropiado
emerg (panic)	Situaciones de emergencia
Alert	Situaciones de urgencia
Crit	Condiciones críticas
Err	Otras condiciones de error
Warn	Mensajes de precaución
Notice	Cosas no usuales que necesitan atención
Info	Mensajes informativos
Debug	Para rastrear

Tabla 3-8 niveles de importancia para los mensajes

El nivel de un mensaje describe su importancia. En el archivo syslog.conf, los niveles indican la importancia mínima que un mensaje debe tener para almacenar la información.

El campo de acción especifica que es lo que se va a hacer con el mensaje. Las posiciones que se tienen son:

Acción	Significado
filename	Escribir mensaje a un archivo en máquina local
@hostname	Enviar mensaje al .syslogd en hostname
@ipaddress	Enviar mensaje a la máquina con el IP address
user1, user2,...	Escribir mensaje a la pantalla del usuario si se encuentra dado de alta en el sistema.

Tabla 3-9 posiciones para los campos de acción

Si se escoge el nombre de un archivo dentro del campo de acción, éste debe ser especificado con su ruta de acceso absoluta. El archivo al que se hace referencia debe existir en la localidad especificada.

Algunas versiones de syslog utilizan una macro llamada m4 en su archivo de configuración. Para la configuración adecuada de las variables de ésta, se deben consultar los manuales del sistema para saber como se van a construir las acciones. Por ejemplo, todas las palabras reservadas deben ser escritas entre comillas:

```
auth.notice ifdef('LOGHOST', /var/log/authlog, @loghost)
```

La instrucción anterior direcciona todos los mensajes a ser grabados a la bitácora /var/log/authlog si se encuentra definida la variable LOGHOST; si no es así, entonces los mensajes serán enviados a la máquina identificada como loghost.

En la tabla 3-10 se presentan algunos programas, junto con su facilidad y nivel en que utilizan el syslog para grabar la información de su funcionamiento.

3.2.4.2 Impresoras

El comando lpr

Adhiere un trabajo a la cola de impresión copiando el archivo al directorio de impresión. Estrictamente hablando, se dedica a poner el trabajo en cola en lugar de imprimirlo; el daemon **lpd** es el que maneja el envío de los archivos a la impresora. Cuando el trabajo es enviado a impresión, se le asigna un número (ID), el cual es utilizado para referenciarlo después con los comandos.

El comando lpq

Proporciona una lista de los trabajos que se encuentran actualmente en la cola. La sintaxis se muestra a continuación:

lpq [opciones] <impresora>

EJEMPLOS

```
#lpq –P pruebaps
Rank    Owner       Job    Files        Total Size
1st     calvin      15     joke.txt     7456 bytes
2nd     hobbes      16     turkey.txt   1234 bytes
```

Programa	Facilidad	Nivel	Descripción
amd	daemon	err-info	Automounter de NFS
Date	auth	notice	Desplegar y poner ftp
Ftpd	daemon	err-debug	Daemon de ftp
gated	daemon	alert-info	Daemon de ruteo
gopher	daemon	err	Servicio de Internet
halt/reboot	auth	crit	Programas de baja del equipo
login/rlogin	auth	crit-info	Programas de login
lpd	lpr	err-info	Daemon de impresión BSD
named	daemon	err-info	Servidor de nombres
nnrpd	news	crit-notice	Lectores de servicios de noticias
ntpd	daemon, user	crit-info	Daemon de sincronización de relojes
passwd	auth	err	Programa para poner passwords
popper	local0	debug, notice	Sistema de correo para MAC/PC
rwhod	daemon	err-notice	Daemon remoto de who
sendmail	mail	debug-alert	Sistema de transporte de correo
su	auth	crit, notice	Programa de sustitución UID
sudo	local2	notice, alert	Programa limitado de su
syslogd	syslog, mark	err-info	Errores internos, problemas de relojes
tcpd	local7	err-debug	TCP-Wrappers para inetd
vixie-cron	cron, daemon	info	Cron estilo ATT para BSD
vmunix	kern	variado	Kernel

Tabla 3-10 programas, facilidades y el nivel de utilización del syslog

El comando lprm

Remueve los trabajos de la cola de impresión. Este comando permitirá a los usuarios eliminar únicamente sus propios trabajos. Solo el superusuario puede eliminar los trabajos de otra persona si ese fuera el caso. La sintaxis se muestra a continuación:

lprm [opciones] <impresora> <trabajo a remover>

El trabajo que se desea eliminar se puede especificar de varias formas: como una lista de identificadores de trabajo y/o por nombre de usuario, o bien, con un simple guión (-) para eliminar todos los trabajos pertenecientes a quien ejecutó el comando. Si fue el superusuario, entonces serán removidos todos los trabajos de la cola de impresión.

EJEMPLOS

```
lprm –P pruebaps 15
```

Con ésta instrucción se remueve el trabajo 15 de la impresora pruebaps.

Para mover un trabajo a otra posición dentro de la cola de impresión, se utiliza la opción **topq** del comando lpc:

lpc> topq <impresora> <trabajo(s)_a_remover>

Los trabajos a remover se pueden especificar de varias formas, como una lista de identificadores de trabajo y/o por nombre de usuario. Este comando moverá los trabajos seleccionados a los primeros lugares de la cola de impresión.

El comando lpc

Lanza la interfaz administrativa para el subsistema de impresión. Se utiliza para llevar a cabo las acciones de administración de los directorios de spool, incluyendo dar de baja una impresora para acciones de mantenimiento, desplegar el estado de una impresora y manipular los trabajos que se encuentran en las colas de impresión. Para invocar la utilería para el control de la impresora, simplemente hay que ejecutar el comando lpc como se muestra en la sintaxis:

lpc

EJEMPLOS

```
# lpc
lpc>
```

en este momento se está ejecutando el comando lpc con su propio prompt y contiene los siguientes comandos internos:

- **status <impresora>** Despliega el estado de las colas de impresión y de la impresora misma.
- **abort <impresora>** Da por terminada inmediatamente cualquier impresión que se encontraba en progreso en ese momento y deshabilita cualquier impresión en la impresora.
- **stop <impresora>** Suspende todo trabajo de impresión en la impresora. Después de que el trabajo se imprime, termina.
- **start <impresora>** Inicializa de nuevo la impresora después de que le fue dado el comando abort o stop.
- **disable <impresora>** Previene a los usuarios de poner nuevos trabajos en la cola de impresión. Solo el superusuario puede añadir nuevos trabajos a la cola de impresión y la impresión continuará, deshabilitar la impresora, esperar a que terminen todos los trabajos pendientes y después parar la impresora. Es la manera más elegante de dar de baja una impresora.
- **enable <impresora>** Permite de nuevo a los usuarios enviar trabajos al directorio de spool. Este comando vuelve a la impresora a la operación normal después de que se le dio el comando disable.

- **down <impresora>** Detiene la impresión y deshabilita la cola de la impresora. Este comando es equivalente a disable más stop.
- **up <impresora>** habilita la cola y empieza el proceso de impresión. Es equivalente a enable más start. Para todos los comandos de lpc, la llave *all* puede ser utilizada para hacer referencia a todas las impresoras conectadas al sistema.

EJEMPLOS

```
lpc> status pruebaps
pruebaps:
        queuing is enabled
        printing is enabled
        no entries
        no daemon present
lpc> disable pruebaps
pruebaps:
        queuing disabled
lpc> stop pruebaps
pruebaps:
        printing disabled
lpc> quit
```

Si no se desea entrar al sistema de administración de lpc, se puede también mandar ejecutar la acción desde el intérprete de comandos:

```
#lpc up pruebaps
```

3.3 PROGRAMACIÓN DEL SHELL

3.3.1 Introducción a los scripts de shell

Un script para el intérprete de comandos o shell es un archivo de texto que contiene una sucesión de comandos de shell. Dicha serie de comandos puede ejecutarse de dos formas esenciales:

- Ejecutando el comando en la línea de ordenes:

  ```
  sh <nombre_archivo>
  ```

  En donde sh es el shell concreto para la que está escrito el script: sh, csh, ksh, tcsh, bash...etc.

- Directamente, dando permiso de ejecución al script, y colocando como primera línea de dicho script el encabezado siguiente:

  ```
  #!/bin/sh
  ```

  En donde, de nuevo, debe indicarse el shell concreto para el cual se escribió el script.

Tradicionalmente los scripts se escriben para el shell sh (Bourne). Así, en lo que sigue, debe entenderse siempre que se está hablando de cómo programar en sh.

Un script en ejecución es un proceso hijo del proceso de shell que lo lanza. Por ello, hereda su entorno (variables de entorno, directorio de trabajo, etc.) y los cambios que en él se hagan no afectan al proceso padre. Por ello, dado el script:

```
#!/bin/sh

cd ..
```

Ejecutado desde la línea de comandos, no produce ningún efecto ``visible'':

- Se crea un proceso cuyo directorio de trabajo es el actual
- Dicho proceso cambia el directorio de trabajo al padre del actual
- El proceso muere

Pueden introducirse comentarios en un script de shell. Un comentario comienza con un carácter "#" ubicado en cualquier lugar de la línea y se extiende hasta el final de la línea.

3.3.1.1 El estado devuelto por un proceso

Todo proceso UNIX puede terminar su ejecución devolviendo un número entero que se denomina estado de salida, el estado es un número entero. Por convenio, un estado de 0 indica una ejecución con éxito del proceso, y un estado distinto de 0 indica que se produjo algún error.

Un script de shell puede devolver un estado de salida con exit. Con exit 3 se devuelve un estado de salida 3, y con exit sin argumento, se devuelve el estado de salida del último comando ejecutado por el script.

3..3.1.2 Encadenamiento de comandos de shell

Puede realizarse con:

| (Pipe)
Convierte la salida estándar de un comando en la entrada estándar del siguiente:

```
who | grep pepe
```

||
Ejecuta el comando de la izquierda, y si falla (termina con un estado distinto de 0), ejecuta también el de la derecha:

EJEMPLO:

```
cat pepe || echo Te equivocas de nombre de fichero
```

&&
Ejecuta el comando de la izquierda, y si tiene éxito (termina con un estado de 0), ejecuta también el de la derecha:

```
cat pepe && echo Bueno, sabias el nombre del fichero
```

3.3.2 Caracteres y variables

3.3.2.1 Caracteres

El shell reconoce a varios caracteres especiales, uno de los más comunes es el carácter asterisco "*". Estos caracteres tienen propiedades especiales por lo que se llaman *metacaracteres*. Existen muchos de estos, y debido a ello, representa una dificultad el indicarle al shell cuando debe ser el carácter tomado como tal, o cuando debe ser tomado como un metacarácter.
La forma más sencilla de impedir que los caracteres sean interpretados como caracteres es encerrándolos entre apóstrofos. Como se ve en el ejemplo:

```
sunserver% echo '*'
*
sunserver%
```

De otra manera si lo hubiésemos tecleado sin los apóstrofos tendríamos el resultado siguiente, algo similar al comando *ls* mencionado anteriormente.

```
sunserver% echo *
archivo1 archivo2 mail
sunserver%
```

Otra posibilidad es el utilizar las comillas, sin embargo esto no es recomendable pues el shell busca dentro de las comillas caracteres especiales como $,'...', y \ porque supone que se realiza algún procesamiento de cadena.

También es posible utilizar la diagonal invertida \ antes de cada caracter que se desee interpretar de manera normal, por ejemplo:

```
sunserver% echo \*
```

Los delimitadores de un tipo protegen a los delimitadores de otro tipo, por lo que no se tiene que cerrar el argumento completo para que la expresión sea reconocida.

```
sunserver% echo a'*'b
a*b
sunserver% echo '*'Y'?'
*Y?
sunserver%
```

Las cadenas que contienen delimitadores pueden contener también el caracter nueva línea.

```
sunserver% echo 'dos
> lineas'
dos
lineas
sunserver%
```

El prompt '>' es el símbolo de espera de órdenes secundario que utiliza el shell cuando espera que un usuario introduzca una entrada para complementar un comando.

El hecho de delimitar un metacarácter evita que el shell trate de interpretarlo como metacarácter y lo interprete como un carácter normal.

El siguiente comando:

```
sunserver% echo a*z
```

Debe desplegar todos los nombres de archivos que comiencen con el carácter *a* y terminen con el carácter *z*, de tal manera que el comando echo no sabe nada de metacaracteres del shell. La interpretación de los metacaracteres la lleva a cabo el shell sin que el comando se de cuenta de ello. Así, si ejecutamos:

```
sunserver% ls a*z
a*z not found          Si el comando no encuentra un archivo correspondiente
sunserver% > ahhhz     Ahora creamos el archivo
sunserver% ls a*z      Ejecutamos nuevamente el comando ls
ahhhz                  Despliega el archivo que concuerda con el patrón
sunserver% ls 'a*z'    Ejecutamos ls con el mismo parámetro entre apóstrofos
a*z not found          ls no interpreta el metacarácter *.
sunserver%
```

El metacarácter # se utiliza para comentarios del shell, si se coloca en alguna parte de la línea, se ignora el resto de la misma, por ejemplo:

```
sunserver% echo prueba # comentario
prueba
sunserver% echo prueba#comentario
prueba#comentario
sunserver%
```

El signo de interrogación identifica a cualquier carácter simple, por ejemplo:

`trabajo?`	Identifica a cualquier nombre de archivo que contenga el prefijo "trabajo" seguido por, exactamente un carácter, por ejemplo trabajo1 pero no trabajo.1.
`*old?`	Identifica a cualquier nombre de archivo que finalice en "old" seguido por, exactamente, un carácter, tal como file.old1.

Los corchetes se utilizan para definir clases de caracteres que identifican a cualquiera perteneciente al conjunto que engloba. Por ejemplo:

`[Jj]urídico` Identifica los nombres de archivo Jurídico o jurídico.

Se puede también indicar un rango o secuencia de caracteres entre corchetes con un signo - . Por ejemplo:

`temp[a-c]` Identifica a *tempa, tempb, tempc*, pero no a *tempd.*

El rango incluye todos los caracteres de la secuencia ASCII desde el primero hasta el último, por ejemplo:

[A-N]	Incluye todos los caracteres de letras mayúsculas entre A y N.
[a-z]	Incluye todos los caracteres de letras minúsculas.
[A-z]	Incluye todos los caracteres de letras mayúsculas y minúsculas.
[0-9]	Incluye todos los dígitos.

3.3.2.2 Variables

El shell usa variables que pueden tomar el valor de una cadena de caracteres para representar tanto valores numéricos como valores de cadena. Hay dos tipos de variables: variables de usuario y variables de shell. Las variables del usuario pueden declararse, inicializarse, leerse y modificarse desde la línea de comandos o desde un programa de shell. El shell declara sus propias variables, algunas de las cuales el usuario puede leer y modificar, otras son simplemente de sólo lectura, las cuales sólo se pueden leer.

3.3.2.2.1 Variables de usuario

Pueden declararse cualquier secuencia de caracteres que no sean espacios en blanco como nombre de un variable. La primera línea de ejemplo siguiente, declara la variable *persona* y la inicializa con la cadena *alex*, el signo igual no debe estar precedido ni seguido de espacio.

```
$ persona=alex
$ echo persona
persona
$ echo $persona
alex
$
```

La segunda línea muestra qué *persona* no representa a *alex* puesto que la cadena *persona* es devuelta como la cadena que es: *persona*. El shell sólo sustituye el valor de una variable cuando va precedido por el signo de pesos ($). El último comando despliega el valor de la variable persona. En el ejemplo que sigue se ilustra cómo marcar una cadena que contenga espacios intercalados para poder asignarla a una variable.

```
$ persona='alex y gina'
$ echo $persona
alex y gina
$
```

3.3.2.2.1.1 Sustitución de variables

El comando echo copia sus argumentos en la salida estándar. La línea de comandos *echo $persona* despliega el valor de la variable *persona*. echo no despliega *$persona* porque el shell no le pasa *$persona* como argumento. Debido al signo ($), el shell reconoce que *$persona* es el nombre de una variable, sustituye el valor de ésta y lo pasa a echo, que a su vez despliega el valor de la variable, no su nombre, sin saber que se le invocó con una variable. El shell habría pasado la misma línea de comandos a echo y éste habría desplegado la misma cadena si se le hubiera introducido la línea *echo alex y gina*. Puede evitarse que el shell sustituya el valor de una variable incluyendo un signo de pesos ($). Las comillas no evitan la sustitución.

```
$ echo $persona
alex y gina
$
```

El comando *shift* efectúa un corrimiento de cada uno de los argumentos de la línea de comandos. El segundo argumento (representado por $2) se convierte en el primero (representado ahora por $1), el tercero en el segundo y así sucesivamente hasta el último. Sólo puede accederse a los nueve primeros argumentos (de $1 hasta $9) de un programa shell. Ejecutar repetidas veces el comando shift brinda acceso a los argumentos que se encuentran de la décima posición en adelante. Sin embargo, no existe un comando unshift (corrimiento hacia atrás) para recuperar los argumentos que ya están disponibles.

EJEMPLO:

```
$ demo_shift erick ruben ramon
arg1= erick arg2= ruben arg3= ramon
arg1= ruben arg2= ramon arg3=
arg1= ramon arg2= arg3=
$
```

En este ejemplo se llamó al programa *demo_shift* con tres argumentos. El programa muestra los argumentos con los que fue ejecutado y los cambia una y otra vez hasta que el tercero (ramon) se convierte en el primero.

3.3.2.2.2 Variables de shell

Las variables del intérprete de comandos constituyen una de las formas en que el intérprete de comandos permite personalizar el ambiente de trabajo. Una variable del shell es un elemento conocido por un nombre que representa algún valor; como es usual en cualquier lenguaje de programación, las variables, como su nombre lo indica, pueden cambiar su valor.

Podemos crear nuestras propias variables del shell, sin embargo esto sólo se utiliza si escribimos programas. Por lo general, utilizamos las variables que vienen con el intérprete de comandos. Podemos establecer algunos de estos valores para modificar el

comportamiento del shell; cada vez que entramos al sistema, el shell inicializa automáticamente las variables siguientes:

Nombre de la variable	Función
HOME=/home/login	Configura el directorio de usuario, que es la localización del directorio de inicio de sesión.
PATH=path	Representa la lista de directorios que el shell examinará para órdenes, se puede configurar la ruta.
PWD=directory	Se configura automáticamente. Esta variable indica el lugar en el que se encuentra el usuario en el sistema de archivos.
LOGNAME=login	LOGNAME se configura automáticamente, al igual que el identificador de entrada al sistema.
SHELL=shell	Identifica la localización del programa que sirve como shell.
MAIL=/var/mail/login	Ruta de direccionamiento de correo, se configura automáticamente
TERM=termtype	Establece el nombre de tipo de terminal utilizada. Generalmente se utiliza como TERM=vt100
PS1=prompt	PS1 es el indicador primario del shell que define el aspecto que tiene el indicador. Si no se configura de algún modo, el indicador será el signo de dólar ($).
TZ	Especifica la zona horaria para la orden date
IFS	Especifica el separador de campos para la línea de comandos

Tabla 3-11 variables inicializadas por el shell durante el arranque.

El shell declara e inicializa las variables mostradas en la tabla arriba. A estas variables se les puede asignar nuevos valores desde la línea de comandos o desde el archivo *.profile* del directorio home.

A continuación se describen más a detalle algunas de las variables de inicio del shell.

HOME: Por omisión, cuando el usuario se da de alta por primera vez, su directorio personal (home directory) es su directorio de trabajo. El administrador del sistema determina el directorio personal del usuario cuando establece su cuenta y almacena esta información en el archivo /etc/passwd. Al entrar al sistema, el shell toma el nombre de ruta del directorio personal del usuario de este archivo y lo asigna ala variable HOME. Cuando no se incluye un argumento, *cd* se convierte en el directorio cuyo nombre está almacenado en la variable HOME, directorio de trabajo. Si se cambia el valor de HOME por un nombre de ruta diferente, *cd* transforma el nuevo directorio en el directorio de trabajo.

EJEMPLO:

```
 $ echo $HOME
/export/home/imc
$ cd
$ pwd
/export/home/imc
$ HOME=/export/home/imc/mail
$ cd
$ pwd
/export/home/imc/mail
$
```

El ejemplo anterior muestra el valor de la variable HOME y el efecto de la utilería *cd*. Después de ejecutar *cd* sin argumento, el nombre de ruta del directorio de trabajo es el mismo que el valor de HOME. Tras asignar a HOME un nombre de ruta diferente, *cd* hace que el directorio de trabajo corresponda al nuevo nombre de ruta.

PATH: Cuando se da un comando al shell, éste busca en la estructura de archivos el programa que se quiere ejecutar. El shell examina varios directorios hasta que encuentra un archivo que tenga el mismo nombre que el comando y para el cual se tenga permiso de acceso de ejecución. La variable PATH controla esta ruta de búsqueda. En general, el primer directorio que examina el shell es el directorio de trabajo. Si el programa no esta allí, la búsqueda continúa en los directorios /bin y /usr/bin, que suelen contener programas ejecutables. Cuando el shell no halla el programa en ninguno de ellos, informa al usuario que no puede encontrarlo.

Si se usa como comando el nombre de ruta absoluto de un programa, el shell no utiliza PATH. Si el archivo ejecutable no se encuentra en el nombre de ruta exacto que se define, el shell informa que no se encuentra el programa. La variable PATH describe los directorios en el orden en que el shell debe examinarlos. Cada uno debe estar separado por dos puntos (:).
El comando siguiente declara algunas de las rutas en que se debe buscar los archivos ejecutables para el shell, la primera es una cadena nula, la cual indica que debe buscarse primero en el directorio de trabajo. Si el shell no encuentra el archivo en el directorio de trabajo, entonces consulta los directorios /export/home/imc/bin y /usr/bin.

```
$ PATH= :/export/home/imc/bin:/usr/bin
```

IFS: Siempre puede emplearse un espacio o un tabulador para separar campos en la línea de comandos. Cuando se asigna a IFS (inter-field-separator: separador de campos) el valor de otro carácter, ese carácter puede utilizarse para separar campos.

El programa *Num_args* informa el número de argumentos con que fue llamado. A continuación, se muestra que fijar IFS puede afectar la interpretación de una línea de comandos.

EJEMPLO:

```
$ Num_args a:b:c
```

Este programa fue llamado con un solo argumento.

EJEMPLO:

```
$ Num_args a:b:c
```

Este programa fue llamado con tres argumentos.

La primera vez que se ejecuta *Num_args*, el shell interpreta la cadena *a:b:c* como un solo argumento. Después de fijar IFS en ":", el shell interpreta la misma cadena como tres argumentos separados.

MAIL: la variable MAIL contiene el nombre del archivo que guarda el correo electrónico. Siempre que entra al sistema un mensaje de correo, este es direccionado hacia la ruta especificada en la variable MAIL. Si se tiene un programa que notifique la llegada de nuevo correo electrónico, éste comprueba el archivo asociado a la variable mail.

PS1: La variable PS1 mantiene la cadena de caracteres que observamos de prompt como indicador primario. El indicador o prompt es la cadena de caracteres que muestra el shell cuando se encuentra listo para recibir una orden.

TERM: La variable TERM se utiliza para identificar el tipo de terminal que estamos utilizando. Los programas que funcionan en la modalidad de pantalla completa como, por ejemplo, el editor vi, necesitan esta información.

TZ: La variable TZ mantiene una cadena que identifica la zona horaria, El programa *date* y algunos otros programas requieren esta información.
El sistema de la computadora controla la hora del sistema de acuerdo con la hora media de Greenwich (GMT). Si la variable TZ se configura a PST8PDT, la hora y fecha se determinan como Hora estándar del Pacífico (PST), ocho horas al oeste de GMT, con soporte para el horario de verano del Pacífico (PDT). El sistema de la computadora cambiará automáticamente entre el horario de verano y el horario estándar.

LOGNAME: La variable LOGNAME mantiene el nombre de entrada al sistema, es decir, el nombre o cadena de caracteres con que el sistema asocia al usuario. La variable LOGNAME identifica al usuario como propietario de sus archivos, como el indicador de los procesos o programas que pudiera estar ejecutando y como el autor de los correos o mensajes que pudiera estar enviando, entre otras cosas.

El comando set

El comando set se utiliza para visualizar el contenido de todas las variables del sistema, la sintaxis es muy sencilla, solo basta teclear la palabra set desde la línea de comando.

EJEMPLO:

```
$ set
HOME=/export/home/imc
IFS=
LOGNAME=imc
MAIL=/var/mail/imc
PATH=/usr/bin:
PS1=$
PS2=>
PWD=/export/home/imc
SHELL=/bin/csh
TERM=vt100
TZ=Mexico/General
USER=imc
$
```

3.3.3 Manejo del editor de textos vi

El editor vi se usa para crear archivos de texto nuevos y modificar los existentes. En este apartado se estudia el funcionamiento de vi, se analizan con detalle muchos de sus comandos y se explica el uso de parámetros para adaptar vi a las necesidades del usuario. Al final del apartado se incluye un resumen de comandos que puede utilizarse como prontuario de consulta.

vi es un editor de textos eficaz (aunque un poco difícil de entender) , interactivo y orientado a pantalla. Aprovecha toda la pantalla de la terminal para desplegar el texto que se está editando. Al usar vi, no es necesario hacer referencia a las líneas por sus números, puede posicionarse el cursor en forma manual en cualquier línea o carácter. vi lleva un registro de lo que está en pantalla y la limpia o refresca cuando es necesario. Este manejo de la pantalla permite a vi desplegar los cambios introducidos en el texto de la manera más eficiente posible y reducir el tiempo de respuesta, en especial con usuarios que acceden al sistema mediante líneas telefónicas lentas.

vi no es un programa para dar formato a texto. No asigna márgenes ni centra títulos, ni tiene las características de un sistema de procesamiento de textos. Para dar formato a un texto editado con vi, se puede utilizar *nroff* u otro programa comercial. Es posible que algún equipo no disponga de vi, pero en general las versiones Berkeley del sistema UNIX, el System V de los laboratorios Bell y aquellos sistemas basados en éstos disponen de él.

3.3.3.1 Buffer de trabajo

El editor vi realiza toda su labor en el buffer de trabajo. Al iniciar una sesión de edición, vi copia el contenido del archivo que se está editando al buffer de trabajo. Durante la sesión de edición, vi hace todos los cambios en esta copia del archivo. vi no modifica el archivo original sino hasta que se da la orden de grabar los cambios hechos. Cuando se edita un archivo nuevo, vi no lo crea hasta que se escribe todo el contenido del buffer de trabajo al final de la sesión. Por lo general, al terminar una sesión de edición, vi vacía automáticamente el contenido del buffer e introduce los cambios en el texto final.

Esta manera de hacer las cosas tiene sus ventajas y desventajas. Cuidado si por accidente se finaliza una edición sin haber escrito todo el contenido del buffer de trabajo, se perderá todo el trabajo. Sin embargo, si por descuido se hacen algunos cambios importantes (como borrar el contenido del buffer de trabajo), puede acabarse la sesión de edición sin introducir los cambios. vi dejará el archivo como estaba antes de iniciar la sesión de edición.

3.3.3.2 Modos de operación de vi

El editor vi es parte de otro editor llamado *ex* e implica a dos de los cinco modos de operación de ex, el modo de comando y el modo de inserción. En el modo de comando, vi acepta los teclazos como comandos y responde a todos los comandos a medida que se introducen. En el modo de inserción, vi acepta como texto los teclazos, desplegando el texto conforme se introduce.

3.3.3.3 Iniciar vi

La manera de editar un archivo o crear uno nuevo con vi es la siguiente:

```
sunserver% vi <nombre_archivo>
```

Donde <nombre_archivo> es el nombre del archivo a modificar o crear. La tabla 3-12 muestra otras maneras de iniciar vi.

Comando para iniciar	Función
vi + n <nombre_archivo>	edita <nombre_archivo> comenzando en la línea n.
vi + <nombre_archivo>	Edita <nombre_archivo> comenzando con la última línea.
vi / <patrón> <nombre_archivo>	Edita <nombre_archivo> comenzando con la primera línea que incluya patrón.
vi –r <nombre_archivo>	Recupera <nombre_archivo> después de una falla del sistema.

Tabla 3-12 Otras maneras de iniciar el editor vi

3.3.3.4 Moviendo el cursor por unidades de medida

Comando	Movimiento del cursor
Espacio, I o flecha derecha	Un espacio a la derecha
h o flecha izquierda	Un espacio a la izquierda
W	Una palabra a la derecha
W	Una palabra delimitada por espacios en blanco a la derecha
B	Una palabra a la izquierda
B	Una palabra delimitada por espacios en blanco a la izquierda
$	Fin de línea
O	Principio de línea
Enter	Principio de la siguiente línea
j o flecha abajo	hacia abajo una línea
k o flecha arriba	Hacia arriba una línea
(	Principio de frase
)	Fin de frase
{	Principio de párrafo
}	Fin de párrafo

Tabla 3-13 Comandos de movimiento

Los comandos mostrados en la tabla 3-13 sólo se emplean desde el modo de comando y pueden ir precedidos por un factor de repetición.

A continuación se muestran algunos comandos para movimiento, en estos se contempla el contenido del buffer de trabajo.

Comando	Movimiento del cursor
control-d	Hacia delante media pantalla
control-u	Hacia atrás media pantalla
control-f	Hacia adelante una pantalla
control-b	Hacia atrás una pantalla
nG	A la línea n
H	Al principio de la pantalla
M	A la parte media de la pantalla
L	Al final de la pantalla

Tabla 3-14 Observación de diferentes partes del buffer de trabajo

3.3.3.5 Adición de texto

Todos los comandos siguientes dejan a vi en el modo de inserción (excepto r). Se debe oprimir la tecla <escape> para volver al modo de comando.

Comando	Inserción de texto
I	Antes del cursor
L	Antes del primer carácter de la línea, que no sea un espacio en blanco
A	Después del cursor
A	Al final de la línea
O	En la siguiente línea hacia abajo (inserta una línea)
O	En la siguiente línea hacia arriba inserta una línea
R	Reemplaza al carácter actual (no se necesita oprimir ESCAPE)
R	Reemplaza caracteres, comienza con el carácter actual (sobrescribe hasta que se oprime escape).

Tabla 3-15 Comandos de edición

A continuación se muestran en la tabla 3-16 los comandos correspondientes al borrado y cambio de texto.

Comando	Función
X	Borra el carácter en curso
D	Elimina el texto del buffer de trabajo
C	Reemplaza con texto nuevo el texto existente
U (undo)	Deshace la última acción. El comando sólo restablece el último texto borrado o modificado por error.
:s/original/reemplazo	Comando de sustitución y reemplazo de cadenas donde original será reemplazado por reemplazo.

Tabla 3-16 Borrado y cambio de texto

Por último, mostramos en la tabla 3-17 una serie de comandos correspondientes a algunas acciones especiales.

Comando	Función
:! comando	Ejecuta el comando del shell
!! comando	Ejecuta el comando del shell y coloca el resultado en el archivo, comenzando por la línea en curso.
:e! archivo	Edita el archivo, descartando los cambios al archivo en curso.
:w nombre	Graba el archivo bajo el nombre indicado
:wq	Graba el archivo con el mismo nombre con el que fue abierto y sale del editor de texto vi. (ZZ es equivalente a wq y a x)
:q!	Sale del editor sin grabar los cambios
:wq!	Guarda y sale del editor. El signo de admiración forza la escritura
:set un	Numera las líneas de texto.
:set list	Despliega caracteres ocultos.
:set all	Despliega la lista completa de opciones en vi
:set noun	Quita la numeración de líneas
:set nolist	Desaparece los caracteres ocultos.

Tabla 3-17 Comandos especiales

3.3.4 Archivos ejecutables

Un archivo ejecutable es aquel que contiene una secuencia de instrucciones o comandos de UNIX escritos, los cuales se ejecutan en orden y tienen el bit de ejecución activado. Un ejemplo de archivo ejecutable en shell puede ser el siguiente archivo:

EJEMPLO:

```
$cat programa
echo 'El numero de personas en el sistema es : '
finger | wc –l
$
```

Este sencillo programa sirve para indicar el número de personas que se encuentran dentro del sistema en el momento de ejecución. Para ejecutarlo se debe escribir el nombre del archivo "programa" en la línea de comando.
Sin embargo si escribimos así simplemente el nombre sin haber activado el permiso de ejecución correspondiente, es seguro obtener el siguiente resultado:

```
$ programa
programa: cannot execute
```

El shell no reconoce a *programa* como archivo ejecutable, por lo que emite un mensaje de error al intentar ejecutarlo. Otros sistemas pueden desplegar un mensaje diferente al mostrado. A continuación se presenta un ejemplo de cómo hacerlo ejecutable

Para ejecutar un archivo de comandos o un programa en shell, se debe otorgar el derecho de ejecución al archivo que contiene el programa. El derecho de ejecución es similar al acceso de lectura o escritura; su función es informar al shell que el usuario, el grupo o los demás usuarios tienen permiso para ejecutar el archivo. La utilería chmod cambia los permisos asociados con un archivo. A continuación se ilustra como *ls* despliega el permiso de programa, antes y después de que el comando chmod otorgue al dueño del archivo el privilegio de ejecución del archivo.

```
$ ls –l programa
-rw[-]rw-r— l haager    36 Feb 13 17:20 programa
$ chmod u+x programa
$ls –l programa
-rw[x]rw-r— l haager    36 Feb 13 17:20 programa
$ programa
El numero de personas en el sistema es:
8
$
```

El primer *ls* muestra un guión como cuarto caracter, lo cual significa que el dueño del archivo **no tiene permiso de ejecución**. El comando *chmod* permite otorgar derecho de ejecución (x) al dueño (u) del archivo mediante el argumento u+x. El último argumento de la línea es el nombre del archivo. El segundo comando *ls* muestra una x como cuarto caracter, lo cual indica que el dueño tiene permiso de ejecución del archivo *programa*. En el último comando se teclea al nombre del programa con lo que se ejecutan los comandos que están escritos dentro del archivo.

3.3.5 Desplegado, aritmética y variables

3.3.5.1 La utilería expr

La utilería expr evalúa la expresión y escribe el resultado en la salida estándar. Toma como argumentos una expresión y efectúa la evaluación de ambas. Las cadenas de caracteres que contengan espacios o caracteres especiales se deben encerrar entre comillas.

La utilería expr se utiliza para:

- Asignar valor a una variable.
- Para contador de iteraciones dentro de un ciclo.
- Para extraer parte de una cadena.
- Para determinar la longitud de una cadena de caracteres.

Los operadores se listan a continuación en orden de procedencia, en caso de que tengan la misma, se encierran entre llaves. Algunos caracteres ya tienen un significado especial para el shell. (|,&,<,>,>>), y por lo tanto se agrega una diagonal invertida para poder utilizarlo dentro de la expresión. De esta manera la sintaxis es la siguiente:

expresión1 \| expresión2

Esta forma devuelve el valor de la expresión1 siempre y cuando no sea nula o cero, de lo contrario regresa la expresión2. En el caso de que las dos sean nulas o cero, regresa cero.

EJEMPLOS:

```
a=999 b=111
expr "$a" \| "$b"
100
```

```
a="" b=111
expr "$a" \| "$b"
200

a="999" b=111
expr "$a" \& "$b"
999
a="" b=200
expr "$a" \& "$b"
0
```

La siguiente sintaxis regresa el resultado de una comparación entera, solo si ambas expresiones son enteras, en caso contrario el resultado es la comparación de dos cadenas. Un valor verdadero se indica con un 1 y uno falso con un 0.

expresión1 {=, \>, \>=, \<, \<=, ¡=} expresión2

EJEMPLO:

```
a=100 b=200 c=100
expr "$a" = "$b"
0

expr "$a" = "$c"
1
```

La siguiente sintaxis de la utilería expr suma o resta el valor entero de las expresiones.

expresión1 {+, -} expresión2

EJEMPLO:

```
a=50 b=200
expr "$a" – "$b"
-150
```

La siguiente sintaxis divide o regresa el residuo de la división, lo anterior se realiza con los valores enteros de ambas expresiones.

expresión1 {*, /, %}

EJEMPLO:

```
a=10 b=20
expr "$a" \* "$b"
200
```

La siguiente sintaxis compara expresión1 con expresión2, esta debe tener un patrón, el cual se busca dentro de la expresión1. Normalmente la respuesta es el número de caracteres que coincidieron.

EJEMPLO:

```
expr man : mano
3
```

3.3.5.2 La utilería bc

La utilería bc es interactiva y permite cálculos con números de punto flotante, la entrada se lee de un archivo especifico, si no existe se lee de la entrada estándar, el resultado de la operación se envía siempre a la salida estándar. La sintaxis es la siguiente:

bc –[lc] archivo de entrada

Soporta los siguientes operadores:

+, -, *, /, %, ^, ++, --, ==, <=, >=, ¡=, <, >, =, +=, -=, *=, /=, %=, ^=

EJEMPLO:
```
bc
2*3
6
```

También soporta los siguientes operandos:
sqrt (expresión) Raíz cuadrada
lenght (expresión) Número de dígitos significativos
scale (expresión) Número de dígitos a la derecha del punto decimal
quit salir de bc

EJEMPLO:
```
sqrt(16)
4

lenght (100/2)
2

scale=10
100/3
33.3333333333
```

3.3.5.3 El comando typeset

Este comando permite definir una variable de tipo entero, lo cual facilita su manejo en las operaciones aritméticas.

EJEMPLO:
```
typest –i num
num=20*3
echo $num
60
```

El comando typeset tiene el alias integer, el cual realiza la misma función de definir una variable de tipo entero.

EJEMPLO:

```
integer num
num=33*3
echo $num
99
```

Una variable no definida entera, se considera como de tipo caracter en el korn shell.

3.3.5.4 Arreglos de variables

El shell permite que un grupo de variables se agrupen dentro de una misma sin perder su identidad, a esto se le conoce como arreglo de variables. A cada variable se le denomina elemento y se hace referencia a ellas mediante un índice. El primer índice es el 0, y el límite máximo de elementos es 512, aún cuando algunas versiones de UNIX soportan más.

```
elemento[0]=uno
elemento[1]=dos
elemento[2]=tres
elemento[3]=cuatro
```

Un arreglo se puede definir de tipo entero mediante el comando typeset.

```
typeset -i elemento[20]
```

Los elementos de un arreglo se pueden inicializar al mismo tiempo:

```
set -A elemento uno dos tres cuatro
```

Para hacer referencia a un elemento se debe indicar el índice, de no hacerlo, se asume el elemento 0. La referencia a un elemento especifico del arreglo se hace con el siguiente formato:

```
${elemento[índice]}
```

```
echo ${elemento[1]}
dos
```

De no utilizarse las llaves { }, los corchetes y el índice [x] se considera una cadena de caracteres que se concatena al elemento 0 del arreglo.

```
echo $elemento[1]
uno[1]
```

Para referirnos a todos los valores de un arreglo, se utilizan los símbolos * y @ y siguen las mismas consideraciones de los parámetros posicionales.

```
echo ${elemento[*]}
```

echo ${elemento[@]}

3.3.5.5 El comando let

Este comando pertenece a korn shell y permite la ejecución de múltiples operaciones en una misma línea.

let operación1 operación2

EJEMPLO:
```
let x=x+1  y=y*3
```

En el ejemplo anterior las expresiones de cada operación no contienen espacios o tabuladores, en caso de incluirlos, las expresiones deben ir entre comillas dobles " ".

EJEMPLO:
```
let "x = x + 1" "y = y * 3"
```

La palabra let se puede sustituir por ((.....)), en este caso la anotación no tiene diferencia si se incluyen espacios o tabuladores, pero tiene la limitación de no permitir múltiples operaciones en una misma línea de comando.

EJEMPLO:
```
((x=x+1))  o  ((X = X + 1))
```

3.3.6 Variables de ambiente y de sistema

Cada vez que trabajamos dentro de un sistema UNIX, es decir, cuando abrimos una sesión dentro de este sistema operativo, trabajamos con un conjunto de variables que definen muchos de los aspectos del entorno de trabajo e influyen de manera importante en la ejecución de los procesos.

A continuación se listan las variables de ambiente utilizadas por el sistema UNIX y la función que cada una de ellas desempeña.

3.3.7 Paso de argumentos

Aunque ya vimos una pequeña introducción al paso de argumentos de shell en un tema anterior, analizaremos un poco mas detalladamente este tema.
La mayoría de los programas en shell interpretan argumentos, de manera que por ejemplo, las opciones y nombres de archivos pueden especificarse cuando el programa se esté ejecutando. Supongamos que deseamos crear un programa llamado "exe" para cambiar el modo de un archivo a ejecutable, de manera que el programa facilite la instrucción y minimice la forma de utilizar chmod en su formato normal. Así el programa se invocaría de la siguiente manera:

```
$ exe <nombre_archivo>
```

De tal manera que <nombre_archivo> representa un parámetro para el programa. Cuando el shell ejecuta un archivo de comandos, cada ocurrencia de $1 se reemplaza por el primer argumento y así sucesivamente hasta el $9. En consecuencia si el script de shell contiene la siguiente línea:

```
chmod +x $1
```

cuando se teclea el comando

```
$ exe programa
```

El shell reemplaza $1 por su primer argumento que contiene el valor de *"programa"*. Observemos la secuencia completa de operaciones:

`$ echo 'chmod +x $1' > exe`	Creamos primero "exe"
`$ exe exe`	Hacemos ejecutable al programa "exe"
`$ echo echo como estas?>hola`	Hacemos el programa "hola" para prueba
`$ hola`	Lo probamos
`hola: cannot execute`	Falla, no se puede ejecutar
`$ exe hola`	Lo hacemos ejecutable
`$ hola`	Probamos nuevamente
`como estas?`	Y funciona!
`$`	

¿Y si deseáramos utilizar más de un argumento? Por ejemplo si quisiéramos hacer que el programa "exe" manejara varios archivos al mismo tiempo podemos colocar nueve argumentos en el programa shell con una instrucción como la siguiente dentro del script.

```
chmod +x $1 $2 $3 $4 $5 $6 $7 $8 $9
```

Sin embargo estaríamos limitados por el argumento numero $9 ya que el $10 sería interpretado nuevamente como $1 con un 0 integrado. Si el usuario de este programa proporciona menos de nueve argumentos, entonces los demás serán cadenas nulas; el efecto es que solo los argumentos que fueron tecleados son pasados al comando chmod a través del shell. Así esto funciona, pero obviamente no es lo es del todo válido, pues falla si se dan más de nueve argumentos.

Para solucionar este problema, el shell proporciona una abreviatura $* que significa "todos los argumentos" la manera correcta de definir exe sería:

```
chmod +x $*
```

Esto funciona sin problemas para cualquier número de argumentos.

3.3.8 Filtros y condiciones

Filtros

Existe en UNIX un gran número de programas que leen la entrada, realizan cambios (proceso) y ofrecen una salida, entre ellos podemos mencionar algunos como *grep* y *tail* para hacer una selección de una parte de la entrada, *sort* para clasificar la entrada, *wc* para contar las palabras contenidas en la entrada, etc. A todos estos programas se les llama "filtros".
A continuación veremos como trabajan algunos de ellos:

3.3.8.1 El comando grep

La sintaxis del comando grep es la siguiente:

grep <patrón> <archivos>

Con lo cual examina los archivos nombrados e imprime en la salida estándar los resultados de haber realizado la búsqueda del patrón. El comando *grep* es de gran utilidad para encontrar ocurrencias de variables en programas o palabras en documentos; también sirve para seleccionar partes de la salida de un programa.

Cuenta con varias opciones como:
-n imprime los números de línea
-v invierte el sentido de la prueba
-y hace que las minúsculas y mayúsculas se acoplen.

EJEMPLOS:

```
$ grep –n variable fuente.c
```

Localiza la variable en el archivo fuente.c

```
$ grep From $MAIL
```

Despliega el encabezado de mensajes en el buzón.

```
$ grep From $MAIL | grep –v jane
```

Todos los encabezados que no provienen de jane

```
$ grep –y jane $HOME/phone-book
```

Encontrar el teléfono de jane en la agenda

```
$ who | grep jane
```

Comprobamos si jane inició sesión

```
$ ls | grep –v txt
```

Archivos que no contienen txt

3.3.8.2 El editor de flujo sed

Ahora veremos lo que es capaz de hacer sed. Su nombre se deriva directamente del editor ed en el cual basa gran parte de su funcionamiento. Básicamente la idea de ed es sencilla y su sintaxis es la siguiente:

sed 'lista de comandos ed' archivos...

Puede leer un renglón a la vez a partir de los archivos de entrada; aplica los comandos de la lista por orden a cada renglón y escribe su forma editada en la salida estándar. De tal manera es posible cambiar UNIX por UNIX(TM) dondequiera que se presente un conjunto de archivos con la sentencia siguiente:

```
$ sed 's/UNIX/UNIX(TM)/g' archivos ...> output
```

Debe quedar claro que *sed* no modifica los archivos de entrada. Escribe los resultados en la salida estándar conservando los archivos originales intactos. *sed* despliega cada renglón automáticamente. Las comillas casi siempre son indispensables porque muchos de los metacaracteres de *sed* significan también algo para el shell.

3.3.8.3 El lenguaje de manejo de procesamiento de patrones awk

Muchas de las limitaciones que tiene sed son remediadas mediante *awk*. La idea es muy semejante a la de *sed*, pero los detalles se basan mas en el lenguaje de programación C que en los comandos de un editor de textos como *ed*. Su sintaxis es igual a la de *sed* y se revisó anteriormente (ver awk en comandos para la manipulación de archivos) .

3.3.8.4 Otros filtros

En este apartado mostraremos la existencia y posibilidades de un rico conjunto de pequeños filtros que ofrece el sistema UNIX. No forman estos el conjunto completo de filtros puesto que se han integrado muchos otros y mejorado algunos de los ya existentes, solo se pretende mostrar las capacidades de algunos de ellos.

El comando sort

Probablemente es *sort* uno de los filtros más útiles, *sort*, como ya lo analizamos anteriormente, sirve para clasificar la entrada por renglones en orden ASCII. Aunque normalmente esta es la función principal que utilizamos de *sort*, hay muchas otras maneras en que uno podría querer organizar los datos. sort proporciona multitud de opciones, por ejemplo:

–f vuelve equivalentes las mayúsculas y minúsculas, eliminando con esto las distinciones entre ellas.
-d (orden de diccionario) ignora con ello todos los caracteres menos las letras, dígitos y blancos en las comparaciones.
-n clasifica atendiendo al valor numérico
-r invierte el sentido de cualquier comparación.
-o especifica un nombre de archivo para la salida, puede ser uno de los archivos de entrada.
-u suprime todos los grupos de renglones, menos uno, que sean idénticos en los campos de clasificación.

EJEMPLOS:

$ ls | sort	Clasifica nombres de archivo por orden alfabético
$ ls –s | sort –n	Clasifica con los archivos más pequeños primero
$ ls –s | sort –nr	Clasifica con los archivos más grandes primero

sort normalmente clasifica un renglón entero, sin embargo se le puede indicar que trabaje solo con campos específicos. La notación –m significa que la comparación omite los primeros campos m; +0 es el inicio de la línea.

EJEMPLOS:

$ ls –l | sort +3nr	Clasificar por contéo de bytes. primero el más grande
$ who | sort +4n	Clasificar por tiempo desde inicio de sesión
$ sort +0f +0 –u archivos	

En este último ejemplo, +0f clasifica el renglón, juntando las mayúsculas y minúsculas, pero los renglones idénticos no necesariamente son adyacentes. En consecuencia, +0 es una llave secundaria que clasifica a los renglones iguales de la primera clasificación y los pone en el orden normal ASCII. Por último, -u excluye los duplicados contiguos. Por eso si tenemos una lista de palabras, renglón por renglón, el comando imprime las palabras únicas.

La opción –u del comando *sort*, fue obtenida basándose en el comando *uniq* (excluye todos los grupos de renglones adyacentes menos uno). El tener un comando especialmente para esta función permite realizar tareas no relacionadas con la clasificación. Por ejemplo, uniq, eliminará los renglones en blanco múltiples, esté clasificada o no su entrada. Las opciones permiten maneras de procesar estas duplicaciones:

-d imrime sólo los renglones duplicados
-u imprime únicamente los que son únicos
-c cuenta el número de ocurrencias de cada renglón

El comando comm

El comando *comm* es un programa que sirve para comparar archivos. En el caso de tener dos archivos de entrada f1 y f2, el comando *comm* imprime como salida tres columnas los renglones que ocurren solo en f1, los de f2 y los que coinciden en los dos. Cualquiera de las columnas de salida puede suprimirse por medio de una opción.

EJEMPLOS:

$ comm –12 f1 f2	Imprime únicamente los renglones que se encuentran en ambos archivos.
$ comm –23 f1 f2	Imprime los renglones que se hallan en el primer archivo pero no en el segundo. Util al comparar directorios.

El comando tr

El comando *tr* transforma los caracteres de entrada. Su función de uso más común es la conversión de mayúsculas y minúsculas.

EJEMPLOS:

$ tr a-z A-Z	Convierte caracteres de minúsculas a mayúsculas
$ tr A-Z a-z	Convierte caracteres de mayúsculas a minúsculas

El comando dd

El comando *dd* sirve principalmente para procesar los datos de cintas procedentes de otros sistemas. Convierte de ASCII a EBCDIC y viceversa. Lee o escribe datos en registros de tamaño fijo con ajuste de blancos, lo cual caracteriza a sistemas que no son UNIX. En la práctica dd se emplea para procesar datos en bruto o sin formato sin importar cual sea su origen.

Por último con el objeto de ejemplificar lo que se logra combinando filtros, analizaremos el siguiente ejemplo con interconexión de procesos, que imprime las 10 palabras de mayor frecuencia de la entrada.

```
$ cat $* | tr –sc A-Za-z '\012' | sort | uniq –c | sort –n | tail | 5
```

En el ejemplo, *cat* reúne los archivos puesto que *tr* se limita a leer la entrada estándar. El comando *tr* comprime en nueva línea los caracteres que no sean letras contiguas, con lo cual convierte la entrada en una palabra por renglón. A continuación las palabras se clasifican y *uniq –c* comprime cada grupo de palabras idénticas en una línea antecedida por un contador, que se transforma en la llave para *sort –n*. El resultado son las palabras únicas en el documento clasificadas por frecuencia creciente. *tail* selecciona las 10 palabras más comunes y con el *5*, las imprime en cinco columnas.

3.3.8.5 Condiciones

Las condiciones del sistema afectan la ejecución de los programas, por eso se utilizan instrucciones de control de condiciones para el buen desempeño de los programas a ejecutar.

Una estructura de control altera el flujo en los programas de shell. El shell proporciona diversas instrucciones simples, como *if*, *case*, *for*, *while* y *until*, sin embargo estas tres últimas las veremos al final.

IF-THEN

A continuación se muestra la estructura de control if-then, donde la instrucción if prueba el estado que devuelve la condición y transfiere el control basándose en ese estado.

if condición
then comandos
fi

EJEMPLO:

```
$ cat if1
echo –n 'palabra 1:'
read palabra1
echo –n 'palabra 2:'
read palabra2
if (test "$palabra1" = "$palabra2")
then echo 'igual'
fi
echo 'fin del programa'
```

IF-THEN-ELSE

La estructura de control *if* ejecuta los comandos que están entre las instrucciones *then* y *else* pasando el control a la instrucción que sigue a *fi*; en caso de un estado falso se ejecutan los comandos que siguen a la instrucción *else*. La sintaxis se muestra a continuación:

```
if condición
then comandos
else comandos
fi
```

EJEMPLO:

```
$ cat if2
echo –n 'palabra 1:'
read palabra1
echo –n 'palabra 2:'
read palabra2
if (test "$palabra1" 0 "$palabra2")
then echo 'igual'
else echo 'Diferente'
fi
echo 'fin del programa'
```

IF-THEN-ELIF

La instrucción *elif* permite construir conjuntos anidados de estructuras *if – then – else*. La sintaxis es la siguiente:

```
if condición
then comandos
elif condición
then comandos
else comandos
fi
```

EJEMPLO:

```
$ cat if3
echo –n 'palabra 1:'
read palabra1
echo –n 'palabra 2:'
read palabra2
echo –n 'palabra 3:'
read palabra3
if (test "$palabra1" = "$palabra2" –a "$palabra2" = "$palabra3")
then echo 'igual:palabra 1, 2 y 3'
elif (test "$palabra1" = "$palabra2")
then echo 'igual: palabras 1 y 2'
elif (test "$palabra1" = "$palabra3")
then echo 'igual: palabras 1 y 3'
elif (test "$palabra2" = "$palabra3")
then echo 'igual: palabras 2 y 3'
else echo 'Diferentes'
fi
$
```

CASE

Cuando se necesita un mecanismo de decisión múltiple la estructura *case* facilita su manejo. Su sintaxis es la siguiente:

case *cadena_de_prueba* in
patrón 1) comando 1;;
patrón 2) comando 2;;
patrón 3) comando3;;
.
.
.
esac

EJEMPLO:

```
$cat case1
echo –n 'Teclee A, B o C:'
read letra
case $letra in
A)  echo 'Usted tecleó A';;
B)  echo 'Usted tecleó B';;
C)  echo 'Usted tecleó C';;
*)  echo 'Usted no tecleó A, B o C';;
esac
```

4.3.9 Ciclos

En algunos casos es necesario que una serie de comandos se repitan un determinado número de veces. Es aquí donde son útiles los ciclos. Las siguientes estructuras permiten que los argumentos introducidos sirvan de control a las veces que debe repetirse el ciclo.

Ciclo FOR – IN

sintaxis:
for *índice_del_ciclo* in *lista_de_argumentos*
do
comandos
done

EJEMPLO:
```
$ cat frutas
for fruta in manzanas naranjas peras platanos
do
echo $fruta
done
```

Ciclo FOR

sintaxis:
for *índice_del_ciclo*
do
comandos
done

EJEMPLO:
```
$ cat prueba-for
do
echo $args
done
```

El valor de la condición determina las veces que los comandos serán ejecutados con las siguientes estructuras.

Ciclo WHILE

sintaxis:
while *condición*
do
comandos
done

EJEMPLO:

```
$cat contador
numero=0
while (test $numero –lt "10")
do
echo –n $numero
numero='expr $numero + 1'
done
```

Ciclo UNTIL

sintaxis:

Until *condición*
do
comandos
done

EJEMPLO:

```
$ cat until 1
nombreclave="gina"
nombre ="ninguno"
echo 'Trate de adivinar el nombre clave!'
echo
until (test $nombre = $nombreclave)
do
echo –n 'Su respuesta:'
read nombre
done
echo 'muy bien'
```

3.4 ADMINISTRACIÓN DE PROCESOS Y UTILERÍAS DEL SISTEMA

3.4.1 Redireccionamiento, concatenación y encadenamiento

El shell puede redireccionar la entrada y la salida estándar de cualquier programa. Realiza este redireccionamiento asociando la entrada o salida estándar del programa con una archivo diferente al dispositivo que representa la terminal del usuario. En este apartado se muestra primordialmente cómo redireccionar la entrada de archivos de texto ordinarios y la salida de éstos. La manera en como se realiza esto, es idéntica para todos los procesos, después utilizaremos la salida de un programa como la entrada de otro y procesaremos entradas de datos.

3.4.1.1 Redireccionamiento

Para direccionar la entrada/salida desde la terminal a algún otro medio, se puede utilizar los símbolos <, >, >>. El símbolo se usa para direccionar la entrada de un archivo hacia un comando o hacia otro archivo. Todos los símbolos de redireccionamiento son utilizados con la siguiente sintaxis:

archivo_o_comando **símbolo** *archivo_de_entrada_o_salida*

EJEMPLO:

```
$ more dir
guadalajara
acapulco
mexico
chiapas
monterrey
$ sort –r < dir
monterrey
mexico
guadalajara
chiapas
acapulco
```

En el ejemplo anterior el archivo *dir* contenía varios nombres de ciudades. Utilizando el símbolo < se direccionó el archivo *dir* al comando para ordenar con *sort –r*, y se obtuvieron los elementos del archivo en orden inverso. El caracter de direccionamiento permite direccionar la salida a un archivo.

EJEMPLO:

```
$ ls > contenido
$ cat contenido
prueba
dir
trabajos
tareas
imágenes
traducciones
IMPORTANTE
contenido
$
```

En este ejemplo la salida del comando *ls* fue direccionada por medio del símbolo > al archivo *contenido*. Es importante tener en mente que si el archivo no existe, éste se crea al direccionar la salida, pero si ya existe, su contenido se sobreescribirá y se perderá la información anterior.

Para direccionar la salida de un proceso a un archivo y agregarla al final del archivo, sin borrar su contenido, se utiliza el símbolo de redireccionamiento >>.

EJEMPLO:

```
$ ls >> contenido
$ cat contenido
prueba
dir
trabajos
tareas
imágenes
traducciones
IMPORTANTE
```

```
contenido
prueba
dir
trabajos
tareas
imágenes
traducciones
IMPORTANTE
contenido
```

En este ejemplo se agregó al archivo contenido la salida del comando *ls*.

3.4.1.2 Concatenación

Concatenar archivos es unir secuencialmente, o extremo con extremo, un archivo con otro para lo cual se utiliza el comando *cat* que despliega un archivo después de otro.

EJEMPLO:

```
$ cat com1
man
who
ls
finger
$ cat com2
stty
clear
mail
$ cat com1 com2 > com3
```

3.4.1.3 Encadenamiento

El shell usa un conductor representado por el símbolo "|" para conectar la salida estándar de un programa directamente a la entrada estándar de otro. Un conducto tiene el mismo efecto como si se redireccionara la salida estándar de un programa a un archivo y después se usara éste como la entrada estándar de otro programa. Prescinde de los mandatos separados y del archivo intermedio. El símbolo de un conducto es una barra vertical "|". A continuación se muestra la sintaxis de una línea de comandos que utiliza un conducto.

```
comando_1 | comando_2
```

Esta única línea de comandos genera el mismo resultado que produciría mediante la secuencia siguiente:

```
comando_1 > temp
comando_2 < temp
rm temp
```

El encadenamiento sirve para crear filtros de información tan complejos como los que analizamos en la sección de "filtros y condiciones".

3.4.2 Administración de procesos

En UNIX, una de las herramientas que hacen al shell tan poderoso para controlar tareas es el manejo de procesos. Debido a que el shell aprovecha el mecanismo de control de procesos es necesario comprenderlo para usarlo eficazmente.

3.4.2.1 Procesos

El sistema operativo UNIX utiliza los procesos como medio para ejecutar programas. Cada comando que se ejecuta crea un proceso. Un proceso es un programa en ejecución. Junto con el programa, el proceso tiene información sobre el usuario y las condiciones del sistema. Cada proceso tiene un número de identificación (process-ID) que se utiliza para distinguirlo de los demás procesos.

3.4.2.1.1 Estructura del proceso

La estructura del proceso es jerárquica, igual que la estructura de archivos. Existen procesos padre, hijos e incluso un proceso raíz. Un proceso padre produce un proceso hijo, el cual, a su vez puede producir otros procesos. En el sistema operativo UNIX, el progenitor y productor de todos los procesos del usuario es el comando *login* para cada terminal. Cuando el usuario entra al sistema se crea un proceso del shell que, por lo general, produce un proceso hijo cuando se ejecuta algún comando tecleado por el usuario.
Mientras el proceso hijo está ejecutando el comando, el proceso padre se mantiene inactivo. Una vez que el proceso hijo termina de ejecutar el programa, muere. Entonces el proceso padre se "levanta" y espera otro comando.

3.4.2.1.2 Identificación de proceso

Cuando se inicia un proceso, le es asignado un número de identificación de proceso (PID). Este número se mantiene sin modificación en tanto el proceso esté vigente. Un proceso hijo tiene un número PID diferente al de su padre. El shell almacena el número PID del proceso que lo está ejecutando en la variable $$. En el siguiente ejemplo se despliega al valor de esta variable.

```
$ echo $$
14437
```

El número que aparece es el PID con el que se ha generado el proceso del shell. Un proceso también tiene información sobre el nombre del comando que se ejecutó, el usuario que ejecutó dicho comando (UID) y el grupo al que pertenece el usuario (GID).

3.4.2.2 Listado de procesos

El sistema operativo mantiene el control de todos los procesos que se ejecutan dentro del shell. Sin embargo, el usuario puede alterar el estado que guardan estos si conoce la información referente a cada proceso. El comando *ps* muestra de forma ordenada y en columnas toda la información acerca de los procesos que está ejecutando el sistema.

El Comando ps

Este comando despliega información sobre el estado de los procesos activos. Su sintaxis es la siguiente:

ps [opciones]

Cuando *ps* se utiliza sin opciones, despliega el estado de los procesos activos como se muestra en el siguiente ejemplo:

EJEMPLO:

```
$ ps
PID    TTY    TIME   COMD
1972   ttyq4  0:00   sh
1984   ttyq4  0:00   ps
```

PID	Esta columna exhibe el número de identificación del proceso.
TTY	Esta columna muestra el número de terminal que controla el proceso.
TIME	Esta columna indica el número de segundos que lleva en ejecución el proceso.
COMD	Esta columna lista el comando con al que se inició el proceso.

Opciones del comando ps

Son varias las opciones que pueden usarse con al comando ps; cada una produce un comportamiento distinto para el comando. La lista de opciones se muestra a continuación.

-a (todos)	Con esta opción despliega información de todos los procesos activos que se ejecutan desde la terminal. Algunos sistemas usan *–e* en lugar de *–a* .
-l (lista larga)	Con esta opción se despliega un informe completo del estado de los procesos.
-x	Esta opción hace que *ps* despliegue información de estado sobre todos los procesos activos, incluso aquellos que no controla la terminal del usuario

3.4.2.3 Suspención de un proceso

Un proceso que está en ejecución puede detenerse o suspenderse momentáneamente si se utiliza la combinación de teclas CTRL-Z (^Z). Al detener un proceso se despliega en pantalla un mensaje indicando que se detuvo el proceso. Los procesos suspendidos se mantienen en ese estado hasta que el usuario los vuelve a poner en actividad. Los procesos pueden reanudarse para que trabajen en primer o en segundo plano. De los procesos en primero y segundo plano hablaremos más adelante.

El comando jobs

Este comando presenta en pantalla una lista con los procesos que se han detenido, así como su respectivo número de proceso. La sintaxis del comando es la siguiente:

jobs [-l]

Opciones
-l Lista la información que el PID presenta con más frecuencia.

3.4.2.4 Primero y Segundo Plano

Los procesos pueden ejecutarse en dos planos, cabe señalar que el usuario puede intercambiar la ejecución de un programa según su conveniencia:

Primer plano (foreground). Se dice que un proceso está corriendo en primer plano cuando es totalmente visible para el usuario y cuando el usuario interactúa con él. Cuando se ejecuta un proceso en primer plano dentro de un shell, el usuario debe esperar a que el proceso termine antes de ejecutar otro proceso.

Segundo plano (background). Los procesos que se ejecutan en segundo plano no se despliegan en pantalla, por lo que es común que el usuario no observe que se están ejecutando. Mientras se desarrolla un proceso en segundo plano, el usuario puede ejecutar otro proceso el cual se ejecuta al mismo tiempo.

El comando fg

Para que un proceso suspendido pueda ejecutarse en primer plano se utiliza el comando *fg* con la siguiente sintaxis:

fg % *número_correspondiente_al_proceso*

El número que se escribe después del símbolo de porcentaje "%" es el que proporciona el comando *jobs.*

EJEMPLO:

```
$ mail juanr
Subject: ^Z
Stopped
$ jobs
[l] + Stopped mail juanr
$ fg % l
mail juanr
Subject:
```

El comando bg

Para mandar un proceso a segundo plano se utiliza el comando *bg* con la siguiente sintaxis:

bg % *número_correspondiente_al _proceso*

Donde el número que se asigna al comando bg es el que corresponde al proceso dentro de la lista desplegada por el comando jobs, sin embargo, ésta no es la única manera de enviar un proceso a segundo plano, pues desde el principio de su ejecución puede hacerse tecleando el comando deseado y en seguida el símbolo &.

EJEMPLO:

```
$ programa > variables &
$
```

En este ejemplo primero se ejecutó programa, luego se direccionó su salida al archivo variables y al final se tecleó el símbolo &, por lo tanto, la ejecución de programa se lleva a cabo en segundo plano. Como observará, puede ejecutarse otro comando sin necesidad de esperar a que termine el proceso que se desarrolla en segundo plano.

3.4.2.5 Comandos útiles para el manejo de procesos

El sistema operativo UNIX permite al usuario modificar el comportamiento de un proceso e incluso terminarlo si ya no es de utilidad. A continuación se presentan dos comandos que pueden servir cuando se trabaja con procesos.

El comando kill

El comando *kill* sirve para terminar la ejecución de un proceso mediante señales de terminación de software. la sintaxis es la siguiente:

kill [-9] *lista_de_PID*

Opciones

-9 Es la única opción que no puede ignorar un proceso

Si se agrega un argumento es posible enviar otra señal distinta. Sólo el dueño del proceso o el superusuario pueden utilizar el comando kill. Cuando termina un proceso, kill despliega un mensaje para indicar la acción.

El comando nice

El comando *nice* se utiliza para otorgar prioridades a los comandos a ejecutar. Su sintaxis es la siguiente:

nice [-0...127] *comando*

Opciones

-0...127 Se puede elegir cualquier número comprendido dentro de este intervalo. Entre más grande sea al número, menor prioridad tendrá el proceso. El C shell define por omisión el número 4 para los comandos.

3.4.3 Sistemas de Archivos Remotos

En un ambiente de red, donde probablemente exista más de un servidor y varias estaciones de trabajo, es necesario distribuir el trabajo y sacar el máximo provecho a todos los equipos enlazados. Hoy en día, el sistema UNIX incluye varios sistemas que permiten a las máquinas compartir y utilizar los recursos que existen en la red, siendo el más utilizado el sistema de archivos de red (NFS: Network File System).

El NFS es un protocolo cliente/servidor que permite compartir archivos y directorios a través de la red. Este protocolo define un servidor NFS que es una máquina que comparte (exporta) sistemas de archivos; por otro lado, el protocolo define clientes NFS, que son máquinas que emplean los sistemas de archivos que comparten los servidores. La relación servidor/cliente NFS se lleva a cabo de manera transparente para el usuario. La ventaja de esta relación es debido a la alta velocidad de transmisión de datos de la red, el usuario no nota la diferencia entre acceso a un archivo remoto y a uno local.

El NFS fue una de las primeras soluciones al problema de compartir archivos remotos y hasta el momento ha sido la más exitosa. Sun Microsystems, compañía que creó el NFS, adoptó la política de ofrecerlo a la industria de cómputo. Muchos proveedores han comprado la licencia y lo han incorporado a sus productos. El NFS es independiente del sistema operativo, por lo que se integra al kernel en la mayoría de los sistemas UNIX. Asimismo, es parte de los protocolos RPC (Remote Procedure Call), estos protocolos diseñados por Sun Microsystems a mediados de 1980 son de dominio público.

3.4.3.1 Como trabaja NFS

El NFS trabaja bajo el modelo cliente/servidor, el servidor comparte sus propios archivos con las demás máquinas de la red, el cliente utiliza los directorios compartidos por el servidor como si fueran suyos; estos pueden ser, por ejemplo, una aplicación para graficar operaciones de manera que el cliente pueda ejecutar esta aplicación aunque el directorio se encuentre fisicamente en el servidor.
Cualquier máquina con un sistema de archivos local puede ser servidor de NFS, a la vez, cualquier máquina puede ser cliente NFS. Todas las máquinas sin disco duro son clientes NFS. El servidor tiene la capacidad de leer o modificar archivos en respuesta a las solicitudes del cliente, lo cual permite, aunque el cliente corra la aplicación, que cualquier cambio se haga en el servidor.

Un servidor NFS asigna archivos locales para que puedan compartirse con otras máquinas en la red. Un proceso en el servidor comparte los archivos y maneja todos los accesos que los clientes ejecuten al archivo compartido; una máquina cliente NFS monta directorios compartidos por el servidor y los trata como si fueran archivos locales.

El servidor no mantiene el estado de los archivos que estén utilizando los usuarios que trabajan en el cliente, es decir, el servidor no lleva un registro de los archivos que se están editando, leyendo o escribiendo en ese momento; el cliente es responsable de este proceso, sin embargo, el trabajo del servidor NFS es revisar que la información se almacene

correctamente. Un servidor NFS también puede ser cliente NFS, sin embargo, no puede compartir directorios que tenga montados como clientes.

EJEMPLO:
Supongamos, la máquina host1 obtiene el directorio /usr/local de la máquina host2. Si una tercera máquina, host3, trata de montar el directorio /usr/local de host1, el acceso le será negado.

```
host2# mount host1:/usr/local
mount: host1:/usr/local: Too many levels of remote in path
mount:gining up on:
/usr/local
```

3.4.3.2 Relación cliente/servidor en NFS

La relación entre clientes y un servidor se establece con programas especiales (conocidos como daemons) de NFS. Estos programas son: biod, nfsd, mountd o rpc.mountd, rpc.lockd y los servicios de red como portmap.

Como se mencionó, el servidor comparte sistemas de archivos o archivos y el cliente los utiliza. El programa *nfsd* controla las solicitudes del cliente, el daemon *nfsd* del cliente establece una comunicación con el daemon *nfsd* del servidor para atender las peticiones del servidor.

El daemon *rpc.mountd* se encarga de dar respuestas a las solicitudes de los clientes, revisa los archivos donde se definen los sistemas de archivos que se comparten (/etc/exports o /etc/dfs/dfstab) en el servidor y dependiendo de las especificaciones de los sistemas de archivos, verifica que los clientes puedan utilizarlos (montarlos).
El daemon *biod* se encarga del control de escritura/lectura del sistema de archivos que comparte un servidor, el daemon *rpc.lockd* se encarga de controlar la ejecución de las aplicaciones remotas de más de un cliente. Sin estos programas especiales, NFS no funciona.

3.4.3.3 Configuración de un servidor y cliente NFS

Servidor

La configuración de un servidor NFS consiste en seleccionar una máquina que contenga el archivo o sistema de archivos que se quiere compartir, en esta máquina se configuran los archivos y se ejecutan los daemon correspondientes al servidor NFS. A continuación se mencionan de manera general los pasos para configurar un servidor NFS. Existe una diferencia entre los sistemas UNIX basados en BSD y los basados en el sistema V.

Pasos
1. Editar el archivo correspondiente al sistema operativo que se utiliza para especificar los archivos o sistemas de archivos que serán compartidos con las máquinas conectadas a red.

Para los sistemas basados en BSD
/etc/exports
directorio –opción,[,opción]

Para los sistemas basados en el sistema V.
/etc/dfs/dfstab
share [-F fstype] [-o opciones] [-d "<descripción>"] <ruta> <recurso>

Donde *"-F fstype"* especifica el tipo de sistema de archivo y *"–d"* se utiliza para detallar información sobre el archivo que se va a compartir.

Nota: Para una referencia completa, consultar los manuales en línea en el sistema correspondiente.

2. Ejecutar el comando respectivo para el sistema operativo que se utilice para exportar los directorios, puede ser el comando *exportfs* o el comando *share*.

3. Ejecutar (o revisar que existan) los daemons del servidor NFS. Estos procesos se ejecutan al inicializarse la máquina. Los daemons son los que se mencionaron en el apartado anterior sobre este mismo tema.

4. Utilizar el comando correspondiente al sistema operativo en uso para verificar que los directorios estén disponibles por la red. En los sistemas basados en BSD utilizar el comando *exportfs*, y en aquellos basados en el sistema V *dfshare*.

Cliente

Un cliente NFS es una máquina que solicita un sistema de archivos al servidor para utilizarlo como si fuera un sistema de archivos local. Si el cliente y el servidor son la misma máquina, entonces el sistema de archivos es local. Si el cliente es una máquina diferente al servidor, entonces esta petición es vía NFS.

Las solicitudes de un cliente a un servidor se hacen por el daemon *rpc.mountd*, el cual se ejecuta por el comando *mount*.

Pasos

1. Editar el archivo correspondiente al sistema operativo en uso para especificar los nombres de servidores y archivos o sistemas de archivos que serán montados al cliente, esto corresponde al archivo /etc/fstab en los sistemas basados en BSD y al archivo /etc/vfstab para los sistemas basados en el sistema V.

2. Crear el punto de montaje. Esto corresponde a crear el directorio en el cual será montado el sistema de archivos.

3. Ejecutar los daemons del cliente NFS o verificar que existan.

3.4.4 Shell Remoto

A veces se puede desear ejecutar una orden en una máquina remota sin abrir sesión en esa máquina, se puede lograr utilizando la orden *rsh* (shell remoto). Una orden *rsh* ejecuta una sola orden en la máquina remota con sistema UNIX sobre una red TCP/IP. Para utilizar *rsh* se debe disponer de una entrada en la base de datos de contraseñas en la máquina remota y la máquina local que se está utilizando debe ser una máquina de confianza para el anfitrión remoto, bien por estar listada en el archivo */etc/hosts.equiv* o por tener una entrada apropiada en el archivo.*rhosts* del directorio propio en la máquina remota. La forma general de una orden *rsh* es:

```
$ rsh host orden
```

Por ejemplo, para producir un listado completo de los archivos que hay en el directorio /home/khr en el host jersey, se utiliza la orden:

```
$ rsh jersey ls –l /home/khr
```

La salida de la orden ls –l en jersey va a la salida estándar de la máquina local. La orden rsh no realiza realmente una apertura de sesión en la máquina remota, en vez de ello, un demonio en la máquina remota genera un shell y luego ejecuta la orden especificada. El tipo de shell generado viene determinado por la entrada en la base de datos de contraseñas de la máquina remota, además se invoca al archivo de arranque apropiado para el shell (es decir, el .profile en la máquina remota si se utiliza el shell estándar).

3.4.4.1 Metacaracteres shell y redirección con rsh

Los metacaracteres shell y los símbolos de redirección en una orden rsh que no estén acotados o escapados son expandidos a nivel local, no en la máquina remota. Por ejemplo la orden:

```
$ rsh jersey ls /usr/bin > /home/khr/list
```

Lista los archivos del directorio */usr/bin* en la máquina jersey, redireccionando la salida al archivo /home/khr/list de jersey, se debe utilizar comillas simples alrededor del signo de redirección > como se muestra en el siguiente ejemplo:

```
$ rsh jersey ls /usr/bin/ '>' /home/khr/list
```

3.4.4.2 Utilización de vínculos simbólicos para ordenes rsh

Si frecuentemente encontramos con que tenemos que emitir órdenes *rsh* sobre una máquina particular, podemos preparar un vínculo simbólico que permita emitir una orden *rsh* sobre esa máquina utilizando simplemente el nombre de la máquina. Por ejemplo, supongamos que ejecutamos la orden:

```
$ ln –s /usr/bin/rsh /usr/hosts/jersey
```

Y coloca el directorio */usr/hosts* en el camino de búsqueda. Así, en vez de utilizar la línea de orden:

```
$ rsh jersey ls /usr/bin
```

Se puede utilizar la línea de orden más sencilla:

```
$ jersey ls /usr/bin
```

Al crear este vínculo simbólico, también puede presentarse remotamente a jersey escribiendo simplemente la orden:

```
$ jersey
```

Esta es una abreviatura de:

```
$ rlogin jersey
```

3.4.5 Protocolo de transferencia de archivos

La copia de archivos desde máquinas remotas es una de las tareas de red más habituales. Puede desearse copiar archivos en máquinas que ejecuten otros sistemas operativos, esto puede hacerse utilizando el protocolo de transferencia de archivos (siempre que la máquina remota soporte el servicio de *ftp*). También puede utilizarse *ftp* para copiar archivos cuando no se conocen los nombres de esos archivos.

La orden ftp implementa el File Transfer Protocol (FTP) que permite realizar sesiones con máquinas remotas. Cuando se emite una orden *ftp* comienza una sesión interactiva con el programa *ftp*. por ejemplo:

```
$ ftp
ftp>
```

Puede visualizarse una lista de las órdenes *ftp* disponibles introduciendo un signo de interrogación ?, o tecleando *help* ante la petición de orden de *ftp*. Puede obtenerse información sobre una orden utilizando la orden *help*, por ejemplo, puede obtenerse información sobre la orden *open* utilizando la línea de orden *ftp*:

```
ftp> help open
```

Para ejecutar una orden ftp sólo se necesita suministrar a ftp tantas letras del nombre de la orden como sean necesarias para identificar unívocamente la orden. Si no se suministran letras suficientes para identificar unívocamente la orden, ftp lo hace saber:

```
ftp n
?Ambiguous command
ftp>
```

3.4.5.1 Iniciar una sesión ftp

Para comenzar una sesión con una máquina remota se utiliza la orden open de ftp. A continuación se muestra un ejemplo de comienzo de una sesión ftp con el host remoto jersey:

```
sunserver% ftp
ftp> open
(to) sunserver
Connected to sunserver.
220 sunserver FTP server (SunOS 5.6) ready.
Name (sunserver:imc): imc
331 Password required for imc.
Password:
230 User imc logged in.
ftp>
```

La primera línea muestra que se ha emitido la orden open de *ftp*, después *ftp* vuelve con el indicador "to"; se introduce *sunserver* como nombre del host remoto, enseguida el servidor FTP responde y pide que se introduzca un nombre de usuario, para esto debemos contar con un identificador, o si el servidor lo permite, podemos utilizar una sesión anónima utilizando la cuenta "*anonymous*". Finalmente se introdujo la cuenta del usuario *imc* y a continuación se pidió que éste se identificara, el servidor verifica el acceso y permite la entrada si es un usuario valido, dejándolo en la línea de comando de *ftp* dentro de la máquina remota.

3.4.5.2 Utilización de órdenes ftp

Una vez que se ha abierto una sesión *ftp* con una máquina remota, se pueden utilizar las muchas y diferentes órdenes *ftp* para efectuar una variedad de tareas sobre la máquina remota. Por ejemplo, se pueden listar todos los archivos que son accesibles con la orden *ls* de *ftp*. También podemos cambiar de directorio utilizando la orden *cd* de *ftp*. Pero no podemos accesar archivos de otro directorio a menos que tengamos acceso permitido a ellos. Cuando se están utilizando órdenes de *ftp*, no se están ejecutando órdenes en la máquina remota directamente, en vez de ello se están dando instrucciones al demonio de *ftp* en la máquina remota.

Se puede escapar del shell de *ftp* y ejecutar una orden shell en la máquina local utilizando un signo de exclamación "!" para salir temporalmente de *ftp*. Para regresar nuevamente se teclea *exit*.

```
ftp> !
```

3.4.5.3 Copia de archivos utilizando ftp

Para copiar un archivo una vez que se ha establecido una conexión ftp se utilizan las órdenes get y put. Antes de copiar un archivo, habría que asegurarse de establecer el tipo de transferencia de archivos correcto. El tipo de transferencia de archivos por omisión es ASCII, para cambiar el tipo de transferencia a binario se utiliza la orden siguiente:

```
ftp> binary
```

para volver al tipo de transferencia de archivos ASCII se utiliza la orden:

```
ftp> ascii
```

Una vez fijado el tipo de transferencia de archivos, pueden utilizarse las órdenes *get* y *put*. Por ejemplo, para copiar el archivos *firma.txt* de la máquina remota *sunserver*, con la cual se ha establecido una sesión *ftp*, se utiliza la orden *ftp* como se muestra a continuación.

```
ftp> get firma.txt
200 PORT command successful.
150 ASCII data connection for firma.txt (148.220.1.2,43305) (856 bytes).
226 ASCII Transfer complete.
local: firma.txt remote: firma.txt
874 bytes received in 0.023 seconds (37.16 Kbytes/s)
ftp>
```

Para copiar el archivo *cuento.txt* a la máquina remota *sunserver*, se utiliza la orden de *ftp* siguiente:

```
ftp> put cuento.txt
200 PORT command successful.
150 ASCII data connection for cuento.txt (148.220.1.2,43311).
226 Transfer complete.
ftp>
```

Cuando se utiliza tanto la orden get como la orden put, ftp informa que la transferencia ha comenzado. También informa cuándo se produce la terminación e indica cuánto ha tardado la transferencia.

3.4.6 Manejo del Correo Electrónico

La amplia disponibilidad y uso del correo electrónico (e-mail) ha producido un efecto impactante en la vida de los usuarios del sistema UNIX. En muchas organizaciones, empresas y universidades, el correo electrónico se ha convertido en una forma estándar de comunicación, de hecho las órdenes relacionadas con el correo electrónico son las únicas que aprenden algunas personas. El correo electrónico permite una comunicación rápida, en la que los mensajes tardan segundos en llegar a su destino, ha modificado las jerarquías de las empresas, permitiendo la comunicación directa entre usuarios de distintos niveles. El correo electrónico permanece en nuestro buzón hasta que queremos leerlo o contestarlo, por lo que la comunicación mediante este método evita los problemas de tiempo real de las llamadas telefónicas.

4.4.6.1 El Programa mail

El programa *mail* del sistema UNIX es el programa de correo más simple y primitivo disponible. Debido a que tiene pocas características complejas, *mail* es especialmente fácil

de utilizar y aprender para un usuario ocasional. La mayor parte de programas de correo electrónico avanzados como *mailx* resultan más fáciles de aprender después de haber utilizado *mail*. Cuando alguien envía correo, el sistema UNIX crea un archivo para mantener los mensajes; si el nombre de usuario es *ana*, entonces la ruta */var/mail/ana* corresponde a la ruta en donde se almacena el correo entrante para *ana*. La orden:

```
$ cat /var/mail/ana
```

Imprime como un flujo continuo el archivo de correo y por lo tanto todo lo que ha recibido.

3.4.6.1.1 Lectura del correo con mail

Si existen mensajes en el archivo de correo, cuando se entra al sistema, se hace una notificación con un mensaje simple como:

```
you have mail (tiene correo)
```

mail proporciona una interfaz de usuario simple que permite leer y tratar los mensajes electrónicos de uno en uno. Simplemente tecleando la orden *mail* en la línea de comando, se imprimiran todos los mensajes, uno a la vez, empezando por los últimos recibidos. Si se quiere visualizar los mensajes en orden cronológico, se utiliza la orden *mail –r* en la línea de comando, lo cual imprimirá los mensajes en el orden de llegada. Después de cada mensaje, el programa *mail* solicita una orden del usuario con el símbolo "?". A continuación se muestra la tabla 4-18 con las órdenes para el programa *mail*.

Orden	Acción
+ o <Enter>	Imprime el siguiente mensaje.
-	Imprime el mensaje anterior.
#	Imprime el número de mensaje actual
a	Imprime los mensajes que llegaron durante la sesión de correo.
d	Borra el mensaje.
dq	Borra el mensaje y termina.
h	Visualiza la ventana de cabeceras en torno al mensaje actual.
h N	Visualiza la ventana de cabeceras en torno al mensaje N.
m persona	Envía este mensaje a persona.
P	Imprime de nuevo el mensaje actual, ignorando la indicación de contenido no imprimible (binario).
q o CTRL-D	Vuelve a colocar el correo no eliminado en /var/mail/you y termina.
u N	NO borra el mensaje N.
r usuarios	Responde al emisor del mensaje y a usuarios; después borra el mensaje.
s archivo	Guarda este mensaje $HOME/mbox se utiliza por omisión si no se especifica archivo.
w archivo	Guarda este mensaje sin su información de cabecera. /mbox es el valor por omisión.
x	Vuelve a colocar el correo electrónico en el archivo y termina.
! orden	Sale al shell y ejecuta orden.
?	Imprime el resumen de órdenes mail.

Tabla 3-18 Lista de órdenes aplicables al programa mail

3.4.6.1.2 Envío de correo con mail

El uso de la orden mail sola significa que se quiere leer el correo. La orden *mail* con una dirección de correo, significa que se quiere enviar correo a esa persona. Por ejemplo:

```
$ mail miguel
```

Se utiliza para enviarle correo al usuario *miguel* del sistema, cuando se especifican varios usuarios, el correo se envía a todos ellos. La siguiente línea de orden envía correo a *miguel, imc* y *fred:*

```
$ mail miguel imc fred
```

Puesto que la orden *mail* acepta entrada estándar, hay dos formas equivalentes de especificar el contenido del mensaje: utilización de la entrada estándar y redirección del shell.

3.4.6.1.2.1 Entrada estándar

Cuando se proporciona a mail como argumento un nombre de presentación (por ejemplo miguel), mail acepta entrada hasta un CTRL-D y después envía esta entrada al usuario nombrado, por ejemplo:

```
$ mail imc
Esto es un correo de prueba.
CTRL-D
$
```

Enviará el correo de prueba al usuario imc.

3.4.6.1.2.2 Redirección

Se puede utilizar el operador de redirección del shell < para hacer que éste proporcione entrada a *mail*. La línea de orden:

```
$ mail imc < memo
```

Tomará el contenido del archivo *memo* y lo enviará al usuario *imc*.

3.4.6.2 El programa mailx

mailx es un programa de correo ofrecido en los sistemas UNIX basados en el Sistema V, *mailx* está basado en el programa *mail* de BSD y proporciona a los usuarios muchas características nuevas, incluyendo un conjunto de instrucciones para manipular mensajes y enviar correo y una interfaz de usuario extendida para correo electrónico. Integra la recepción y la contestación de correo y proporciona docenas de características y opciones. Por ejemplo *mailx* permite inspeccionar el remitente y la materia de un mensaje antes de decidir leerlo; esto permite conmutar rápidamente entre lectura, envío y edición de

mensajes de correo, también permite personalizar la forma en que opera el correo para ajustarlo a nuestras preferencias.

3.4.6.2.1 Lectura de correo con mailx

Cuando se lee correo con *mail*, los mensajes aparecen en su totalidad. *mailx* permite examinar los mensajes antes de leerlos mediante la visualización de las cabeceras de los mensajes. *mailx* proporciona una pantalla introductoria que identifica al remitente, la fecha, el tamaño y la materia de cada mensaje. En base a esta información, podemos seleccionar mensajes específicos para leerlos o ignorarlos. Los que sigue es un ejemplo de una pantalla *mailx*:

```
sunserver% mailx
mailx version 5.0 Tue Jul 15 21:29:48 PDT 1997  Type ? for help.
"/var/mail/imc": 3 messages 1 unread
U>1 The Driver Guide  Tue    Aug 24 19:23  51/1890  Your Driver Guide Members
O 2 Romaine           Wed    Aug 25 14:00  168/5436 [English-L] The English
O 3 Developer.com     Wed    Sep 8  21:29  246/10910 DEVELOPER.COM JOURNAL
?
```

La visualización de *mailx* proporciona mucha información. La primera línea identifica la versión del programa que se está utilizando, visualiza la fecha y recuerda que existe ayuda disponible tecleando ?. La siguiente línea visualiza el nombre del archivo que está leyendo el programa *mail* (/var/mail/imc), el número de mensajes (3) que tiene y su estado (no leídos). Las restantes líneas contienen información de cabecera de los mensajes.

El símbolo > apunta al mensaje actual, el primer mensaje identificado por el número 1 y marcado con una U indica que no ha sido leído. Este mensaje procede de "The driver guide", fue recibido el día martes 24 de agosto a las 19:23 horas.

El signo ? en la parte inferior de la visualización es el signo de solicitud de órdenes *mailx*. En este punto *mailx* espera una orden. Se puede leer cualquiera de los mensajes simplemente tecleando el número correspondiente después del símbolo de interrogación.

La tabla 3-19 muestra una serie de órdenes para *mailx*:

3.4.6.2.2 Envío de correo con mailx

mailx trabaja de forma similar a *mail* en cuanto al envío de mensajes. Las similitudes facilitan pasar del uso de *mail* al de *mailx*. Para enviar un correo a alguien, se utiliza la orden *mailx* seguida por el nombre de usuario o la dirección de correo completa, por ejemplo:

```
sunserver% mailx miguel
Subject: Notificacion
Miguel:
Recibi tu mensaje. Espero poder asistir

CTRL-D
```

Orden	Abreviación	Significado
!		Ejecuta una orden del sistema UNIX.
#		Ignora el resto de la línea (comentario).
=		Imprime el número de mensaje actual.
delete	d	Borra el mensaje actual.
dp		Borra el mensaje actual e imprime el siguiente mensaje.
edit	e	Edita el material tecleado en este mensaje.
exit		Sale, abandona mailx y guarda mailfile exactamente como estaba al inicio de la sesión.
from	f	Visualiza las cabeceras de los mensajes especificados.
headers	h	Muestra las cabeceras actuales.
Help	?	Imprime un breve resumen de las órdenes mailx.
mail	m	Envía este mensaje al usuario especificado.
next	n	Visualiza el siguiente mensaje.
print	p	Visualiza este mensaje en la pantalla.
quit	q	Sale de mailx, borrando, guardando y así sucesivamente, todos los mensajes que emitimos por las ordenes.
reply	r	Responde a un mensaje, enviando respuesta al emisor originy al resto de los receptores del mensaje.
Reply	R	Responde a un mensaje, enviando respuesta sólo al autor del mensaje original.
save	s	Guarda el mensaje en mbox (por omisión) o en un archivo especificado.
Save	S	Guarda el mensaje en un archivo con el nombre del emisor del mensaje.
top	to	Visualiza n líneas superiores de esta cabecera de mensaje.
type	t	Visualiza el mensaje sobre la pantalla.
undelete	u	Restaura los mensajes borrados.
versión		Imprime la versión actual del programa mailx.
visual	v	Utiliza el editor vi (visual) para editar.
write	w	Guarda el mensaje sin información de cabecera de mensaje.
xit	x	Sale, abandona mailx y guarda mailfile exactamente como estaba al comienzo de la sesión.
z+		Visualiza la siguiente pantalla de cabeceras.
z-		Visualiza la pantalla anterior de cabeceras.

Tabla 3-19 Órdenes para mailx

3.5 ADMINISTRACIÓN DEL SISTEMA UNIX

Cada propietario de una computadora debe estar al tanto de las tareas de administración del sistema. Incluso aquellos que usan sistemas simples como Windows o DOS tienen responsabilidades de administración del sistema, necesitan instalar el software del sistema, los programas que usarán y los dispositivos hardware como impresoras, unidades de disco, escáners. También necesitan eliminar archivos innecesarios, defragmentar y optimizar discos fijos y hacer copias de seguridad de los datos regularmente.

Esto mismo sucede en cualquier sistema UNIX, tanto si es una computadora personal como un mainframe. La extensión de la administración que deba hacerse, dependerá de cómo se utiliza la computadora. Si se tiene una estación de trabajo personal o se es el único usuario de la computadora, la administración inicial puede ser tan simple como conectar el hardware de la computadora, instalar el software y definir algunos pocos elementos básicos como el nombre del sistema, la fecha y la hora.

Dado que el sistema UNIX es un sistema operativo multiusuario, puede prepararse la computadora para que utilice mucha gente. Como administradores, debemos asignar nombres de usuario, contraseñas y directorios de trabajo para cada usuario. Probablemente habrá necesidad de conectar terminales adicionales para que varios usuarios puedan trabajar al mismo tiempo.
También se debe proteger la información en la computadora. Para hacer eso se necesita controlar el espacio disponible en disco y la velocidad de procesamiento, proteger frente a violaciones de seguridad y hacer copias de seguridad de los datos regularmente. Dependiendo de la clase de trabajo que se esté realizando en la computadora, puede necesitarse añadir impresoras, redes y otros periféricos hardware.

Aunque no se necesita ser un gurú de UNIX para hacer administración del sistema básica, se necesita estar familiarizado con las características básicas de UNIX y tener alguna destreza para editar y emitir órdenes. Hay mucho que aprender, pero ser un administrador del sistema competente es un papel importante, merece la pena el esfuerzo necesario para aprender a administrar este sistema operativo.

3.5.1 Instalación del sistema operativo

En esta sección se verán las bases para la instalación de un sistema UNIX, sin embargo nos limitamos al análisis de aspectos teóricos y no prácticos, pero se recomienda practicar con la instalación de un sistema UNIX en PC como lo es Linux para completar el proceso de aprendizaje del alumno.

3.5.1.1 Tipos de configuración de una máquina

3.5.1.1.1 Servidores

Un servidor es una máquina o un proceso que proporciona servicios a las máquinas de la red. Generalmente, un servidor es una máquina con un CPU poderoso, mucha memoria y discos veloces, esto no implica que no podamos instalar un servidor en un equipo con recursos limitados, pero su desempeño será mucho menor al de un equipo con bastos recursos.

3.5.1.1.2 Clientes

Un cliente es una máquina o un proceso que utiliza los recursos de los servidores de la red.

3.5.1.1.3 Máquinas standalone

Una estación de trabajo *standalone* es una máquina que no requiere ningún servicio de la red para comenzar a trabajar. Por ser independiente de la red, una estación *standalone* tiene su propio disco duro y posiblemente su dispositivo para realizar respaldos. No necesita ningún servicio de la red, aunque la mayoría de las estaciones *standalone* están conectadas a la red. Una estación de trabajo de tiempo compartido es generalmente una estación *standalone* con terminales conectadas a sus puertos seriales.

3.5.1.1.4 Clientes diskless

Una máquina diskless es una estación gráfica de trabajo que no tiene disco duro propio, por lo que tiene que solicitar sus sistemas de archivos básicos a algún servidor de la red. Normalmente conocidas como terminales X.

3.5.1.1.5 Clientes dataless

Un cliente dataless es una máquina que tiene un disco duro local pequeño que utiliza para el sistema de archivos root y el área de swap. Los sistemas de archivos /usr y /home los obtiene de algún servidor de la red.

3.5.1.1.6 Servidor de dataless y diskless

Un servidor de esta naturaleza debe tener grandes capacidades de almacenamiento, ya que además de contar con sus propios sistemas de archivos, tendrá que almacenar los de los clientes.

3.5.1.2 Requerimientos de configuración de una máquina.

3.5.1.2.1 Máquinas standalone

Requerimientos básicos de hardware.
- Monitor y teclado.
- CPU y memoria.
- Disco duro de al menos 200 Mb aproximadamente.
- Dispositivo de respaldo (opcional)

Requerimientos de sistemas de archivos
- Sistema de archivo raíz (/).
- Área de swap
- Sistema de archivos /usr
- Sistema de archivos /home o /export/home
- Sistema de archivos /opt

3.5.1.2.2 Clientes diskless

Requerimientos básicos de hardware
- Monitor y teclado
- CPU y memoria
- Hardware Ethernet

Requerimientos de sistemas de archivos
- Todos sus sistemas de archivos los obtendrá a partir de la red. Estos son: / (root), /usr, /home y /opt.

3.5.1.2.3 Clientes dataless

Requerimientos básicos de hardware
- Monitor y teclado
- CPU y memoria
- Hardware Ethernet
- Disco pequeño

Requerimientos de sistemas de archivos
- Sistemas de archivos
- Área swap
- Los sistemas de archivos /usr, /home, y /opt los obtendrá a través de la red de algún servidor.

3.5.1.2.4 Servidores diskless y dataless

Requerimientos básicos de hardware
- Monitor y teclado
- CPU y memoria
- Hardware Ethernet
- Disco duro mayor de 300 Mb
- CD-ROM
- Dispositivo de respaldo

Requerimientos de sistemas de archivos
- Sistema de archivos /
- Área de swap
- Sistema de archivos /usr
- Sistema de archivos /export/home o /home
- Sistema de archivos /opt
- Para los clientes tendrá que tener /export/root, /export/usr y /export/swap

3.5.1.3 Información básica para la instalación del sistema operativo

El proceso de instalación del sistema operativo UNIX puede variar según el fabricante. Por esta razón, la información básica para la instalación del S.O. también puede diferir.

La tabla 3-20 muestra la información básica con que cualquier equipo UNIX conectado en red deberá contar al momento de instalación del sistema operativo:

3.5.1.4 El proceso de instalación

El proceso de instalación del sistema operativo varía de acuerdo con el sistema operativo que se esté manejando, para ello se deberá tener conocimiento suficiente del hardware y el entorno en el que se encuentra la máquina a la que se le instalará UNIX. Para esta sección se recomienda consultar la guía de instalación del fabricante.

INFORMACIÓN	EJEMPLO
Tipo de configuración	Standalone
Nombre de la máquina	azteca
Dirección IP	148.202.3.1
Máscara para la subred	255.255.255.0
Servicio de nombres	Nis
Dominio	staff.udg.mx
Nombre del servidor de nombres	cosmos
Dirección IP del servidor de nombres	148.202.3.4
Software que se instalará	Básico
Particiones (opcional)	/,swap,/export/home y /usr

Tabla 3-20 Información necesaria para la instalación de un equipo UNIX

3.5.2 Comprensión de la administración del sistema

Un sistema UNIX debe tener una o más personas designadas como administradores del sistema para gestionar el sistema y prever su rendimiento. El administrador del sistema tiene la responsabilidad del funcionamiento adecuado del sistema, de saber a quién llamar si no puede resolver los problemas internamente y de saber cómo proporcionar recursos hardware y software a los usuarios.

Un sistema UNIX precisa una configuración inicial y luego atención continua para asegurar que el sistema se mantiene efectivo, fiable y eficiente para todos los usuarios. El administrador del sistema es la persona responsable de atender las necesidades del sistema. Por ello es responsable de muchas tareas diversas.

3.5.2.1 La importancia de una administración adecuada

Todos sistemas UNIX son distintos de una forma u otra y cada sistema individual UNIX es distinto en la forma en que debe administrarse. Las tareas de administración varían, dependiendo, entre otras cosas, del número de usuarios a administrar, los tipos de periféricos conectados, las conexiones de red y el nivel de seguridad necesaria.

Un administrador del sistema, ya sea solo, o ayudado por otros, tiene que proporcionar a los usuarios del sistema un entorno seguro, eficiente y fiable. Tiene el poder y la responsabilidad de establecer y mantener un sistema que proporcione servicio fiable y efectivo. En un entorno multiusuario hay un cierto número de objetos y prioridades. El administrador ejerce el poder y la responsabilidad necesarios para proporcionar un sistema que funcione correctamente.

La delegación de las responsabilidades de administración varía de un sistema a otro. En sistemas grandes las tareas de administración pueden dividirse entre varias personas, por otro lado algunos sistemas pequeños ni siquiera necesitan un administrador de tiempo completo; en tales sistemas se le asigna un simple usuario la tarea de administrador. Si se está trabajando en un entorno de red, el sistema puede administrase a través de la red por un administrador de red.

Todos los sistemas UNIX tienen un solo usuario que puede realizar prácticamente cualquier operación en el servidor, a este usuario se le denomina *superusuario* y tiene un nombre especial de entrada al sistema llamado *root*. Cuando el usuario root entra al sistema, lo hace en el directorio raíz del sistema de archivos o en un directorio especial llamado /root.

El administrador del sistema entra como superusuario para realizar tareas privilegiadas; para el trabajo normal en el sistema, el administrador entra como usuario normal. El nombre de entrada del superusuario "root" se utiliza solo para propósitos limitados y especiales, el número de usuarios que pueden entrar como root deben ser los menos posibles. Cuando una persona entra como root, se convierte en superusuario y tiene privilegios absolutos en el sistema, con este privilegio puede cambiar los atributos de cualquier archivo, iniciar el sistema, hacer copias de seguridad de los datos del sistema y muchas tareas más.

El administrador tiene que conocer muchos de los aspectos técnicos del sistema, también tiene que saber cuáles son las necesidades de los usuarios así como cuál es el propósito primario del sistema. Cualquier sistema informático es un recurso finito, deben establecerse y hacerse cumplir normas que regulen su uso. Así, el administrador tiene un papel técnico y normativo; ese papel, combinado con la posibilidad de realizar virtualmente cualquier cosa, hace necesario una persona responsable, habilidosa y diplomática para el papel de administrador.

3.5.2.2 Tareas administrativas comunes

La descripción exacta del trabajo del administrador del sistema frecuentemente depende de la organización interna. Un administrador del sistema puede encontrase envuelto en una amplia variedad de actividades, desde establecer las normas para instalar software, conectar dispositivos, terminales, etc., sin embargo hay una serie de tareas que todos los administradores tienen que gestionar:

- *Administración de usuarios*: Dar de alta a usuarios, darlos de baja y modificar sus posibilidades y privilegios.
- *Configuración de dispositivos*: Hacer disponibles y compartir dispositivos como, por ejemplo, impresoras, terminales, módems, unidades de respaldo, etc.
- *Creación de copias de seguridad*: Programar, hacer y almacenar copias de seguridad para poder restaurar si se dañan o pierden los archivos del sistema.
- *Apagado del sistema*: Apagar el sistema de un modo ordenado para evitar inconsistencias en el sistema de archivos.
- *Formación de usuarios*: Proporcionar directa o indirectamente formación a los usuarios de modo que puedan utilizar el sistema de forma efectiva y eficiente.
- *Aseguramiento del sistema*: Evitar que los usuarios interfieran unos con otros a través de acciones accidentales y deliberadas.
- *Registrar los cambios del sistema*. Mantener un libro para registrar cualquier actividad significativa que se refiera al sistema.
- *Asesoría a los usuarios*: Actuar como experto para ayudar a los usuarios.

3.5.3 Iniciar y dar de baja el sistema

UNIX es un sistema operativo complejo, y arrancarlo o darlo de baja es más complicado que el simple hecho de presionar el botón de apagado/encendido. Ambas operaciones pueden realizarse correctamente si el sistema está en condiciones adecuadas.

"Bootear" es la palabra utilizada para identificar el hecho de "Arrancar una computadora". Las facilidades normales de un sistema operativo no están disponibles durante el proceso de inicio del equipo; durante el proceso de inicio, el kernel es cargado dentro de la memoria e iniciada su ejecución, una variedad de tareas de inicialización son realizadas y posteriormente el sistema queda disponible a los usuarios.

El tiempo de boot es un período especial de vulnerabilidad. Errores en los archivos de configuración, dispositivos no reconocidos o perdidos pueden ser un problema y sistemas de archivos dañados pueden completamente impedir que una computadora pueda levantar correctamente. La configuración del arranque del sistema es una de las primeras tareas que un administrador debe desempañar en un nuevo sistema, desafortunadamente, esta es una de las tareas más difíciles y requiere estar familiarizado con muchos otros aspectos del sistema UNIX.

3.5.3.1 Arranque manual y automático

La mayoría de los sistemas pueden arrancar en cualquier modo, manual o automático. En el modo automático, el sistema ejecuta un procedimiento completo de arranque por si mismo, sin ninguna asistencia externa. En modo manual, el sistema llama el procedimiento automático de arranque hasta un cierto punto, pero después pasa el control al operador antes de que la mayoría de los scripts de inicialización se ejecuten; en este momento la computadora está en modo "usuario-único". La mayoría de los procesos del sistema no están corriendo y otros usuarios no podrán entrar.

El arranque automático es utilizado casi de manera exclusiva, un procedimiento de arranque típico para una máquina moderna es accionar el botón de encendido y esperar a que el sistema se ponga en línea por si solo; no obstante, es importante comprender el procedimiento de arranque automático y conocer como se realiza un arranque manual. Nosotros como administradores debemos arrancar manualmente cuando hay un problema que interrumpa el arranque automático, tal como un sistema de archivos corrupto o una configuración no apropiada para la interfaz de red.

Pasos en el proceso de arranque

Un proceso de arranque típico consiste en seis distintas fases:

- Cargar e inicializar el kernel.
- Detección y configuración de dispositivos.
- Creación de procesos del sistema automáticos.
- Intervención del operador (usuario-único)
- Ejecución de scripts para inicio del sistema.
- Operación Multi-usuario

Un administrador tiene poco control sobre la mayoría de estos pasos. La mayoría de las configuraciones de arranque se logran en la edición de los scripts de inicio del sistema.

3.5.3.1.1 Inicialización del kernel

El kernel de UNIX es un programa que trabaja por si solo, y la primera tarea durante el arranque es cargar éste dentro de la memoria de tal manera que pueda ser ejecutado. La ruta en donde se localiza el kernel es diferente en cada sistema y depende del vendedor, pero es común encontrarlo en los directorios /unix o /vmunix.

La mayoría de los sistemas implementa un proceso de carga en dos etapas; durante la primer etapa, un programa de arranque pequeño es leído dentro de la memoria desde un disco o cinta, este programa permite que el kernel sea cargado. Todo esto ocurre fuera del dominio de UNIX, tal que no hay una estandarización en absoluto entre sistemas.

3.5.3.1.2 Configuración de hardware

Una de las primeras tareas del kernel es checar el ambiente externo de la máquina para detectar que hardware está presente. Cuando construimos un kernel para nuestro sistema decimos que tipo hardware o dispositivos esperamos encontrar, así, cuando el kernel inicia su ejecución, este intenta localizar e inicializar cada dispositivo que especificamos. La mayoría de los kernels imprimirán una línea de información acerca de cada dispositivo encontrado.

La información de dispositivos proporcionada en tiempo de configuración del kernel es frecuentemente mal especificada, en estos casos, el kernel intenta determinar que otra información es necesaria para probar los dispositivos y preguntar la información apropiada para los controladores. Los controladores de dispositivos que no son reconocidos o no responden a las pruebas, serán desactivados. Aún si un dispositivo es conectado después al sistema, este no será accesible al sistema hasta que la máquina sea nuevamente inicializada.

3.5.3.1.3 Procesos del sistema

Una vez que la inicialización básica está completa, el kernel crea algunos procesos "espontáneos" en el espacio del usuario. El número y naturaleza de estos procesos varía de sistema a sistema. En un sistema basado en BSD, hay tres:

- swapper – proceso 0.
- init – proceso 1.
- pagedaemon – proceso 2.

El número exacto de procesos espontaneos varía en los sistemas V:

- sched – proceso 0.
- init – proceso 1.
- varios manejadores de memoria (excepto en Solaris).

De estos procesos, solamente *init* es un proceso de usuario; los otros son actualmente porciones del kernel que han sido iniciados para observar los procesos por razones de ordenamiento cronológico.

Hasta este momento, el papel del kernel en el proceso de inicialización está completo, sin embargo, ninguno de los procesos que manejan operaciones básicas (como es aceptar logins en terminales) ha sido creado, ni tampoco han sido inicializados los daemons en UNIX. Todas estas tareas son tomadas cuidadosamente por el *init*.

3.5.3.1.4 Intervención del operador

Si el sistema es levantado en modo usuario-único, *init* es notificado de eso, tal que éste inicializa a través de una señal que pasa al kernel a través de la línea de comando. Durante el arranque en modo usuario-único, *init* simplemente crea un shell en la consola del sistema y espera a que este termine antes de terminar con el resto del proceso de inicialización. El shell de usuario-único es siempre el Bourne shell (sh), no el C shell (csh), éste corre como root, o bien, superusuario.

Desde el shell de usuario-único el operador puede ejecutar comandos de la misma manera que cuando entra el sistema iniciando en modo normal, sin embargo, solo la partición de root es comúnmente montada; el operador puede montar otros sistemas de archivos para manejar sus programas que no se encuentren en los directorios /bin, sbin y /etc. Los daemons comúnmente no corren en modo usuario-único, tal que los comandos que dependen del servidor de procesos (por ejemplo mail) no trabajarán correctamente.

El comando *fsck*, el cual checa y repara sistemas de archivos, normalmente corre durante un arranque automático. Cuando el sistema es levantado en modo usuario-único, *fsck* puede correrse manualmente.

3.5.3.1.5 Scripts de inicio

El siguiente paso en el procedimiento de inicio es la ejecución de scripts o guiones de inicio. Estos scripts son realmente solo scripts de shell, e init corre sh para interpretarlos. La localización exacta, contenido y organización del los scripts varía de sistema a sistema.

Hay dos formas predominantes para organizar scripts de inicio, en los sistemas BSD, los scripts se mantienen en el directorio /etc y tienen nombres que inician con el prefijo "rc". En los sistemas V, los scripts se mantienen en /etc/init.d, y las ligas dentro de los directorios /etc/ etc/rc0.d, /etc/rc1.d y así sucesivamente.

Algunas de las tareas que son frecuentemente ejecutadas en los scripts de inicialización son:

- Establecer el nombre de la computadora.
- Establecer la zona horaria.
- Checar los discos con fsck (solo en modo automático).
- Montar los discos de sistema.
- Remover archivos del directorio /tmp.

- Configurar las interfaces de red.
- Inicializar los daemons y los servicios de red.
- Iniciar las cuentas de usuario.

La mayoría de los scripts son bastante concisos e imprimen una descripción de cualquier cosa que están haciendo. Esto es una gran ayuda si el sistema falla a medio camino durante el arranque o si estamos intentando localizar un error en los scripts.

3.5.3.1.6 Operación multi-usuario

Después de que los scripts de inicialización han sido corridos, el sistema esta completamente en modo operacional, excepto por el hecho de que ningún usuario puede entrar. Para que los logins sean aceptados, debe haber un proceso getty que espera esto. Después de que los scripts de inicio han sido corridos, init comienza este proceso getty, completando el proceso de inicio. *init* siempre representa un rol importante, aún después de que se ha completado el proceso de inicio.

En sistemas BSD, init tiene solamente dos estados: usuario-único y multi-usuario. En un sistema moderno basado en sistema V, init tiene niveles de trabajo, un usuario-único y muchos multi-usuario que determinan cual de los recursos del sistema están disponibles para cada uno.

3.5.3.2 Reinicializar y dar de baja el sistema

En una PC, reiniciar el sistema operativo es el primer camino a seguir para resolver cualquier problema ocasionado, en un sistema UNIX, es mejor pensar primero en resolver el problema y reiniciar después. En UNIX los problemas suelen ser delicados y más complejos, tal que reiniciar el sistema es una manera efectiva de resolver en un bajo porcentaje de los casos. Los sistemas UNIX además toman gran tiempo en reiniciar y muchos usuarios pueden desesperarse.

Podemos reiniciar cuando se adiciona una nueva pieza de hardware, o cuando existe hardware que está en conflicto y no se puede reestablecer. Si modificamos un archivo de configuración que sólo se lee en tiempo de inicio, entonces debemos reiniciar para que los cambios tomen efecto, y obviamente, si el sistema está trabado, tal que no podamos entrar para crear un diagnostico propio del problema, entonces no tenemos otra alternativa que reiniciar.

Hay diferentes formas para dar de baja o reiniciar un sistema UNIX. Estas son:

- Desconectar la corriente.
- Utilizar el comando shutdown.
- Utilizar los comandos halt y reboot (BSD).
- Enviar a init una señal TERM.
- Utilizar telinit para el nivel de ejecución en init (Sistema V).
- Eliminando init.

3.5.3.2.1 Desconectando la corriente eléctrica

Aún en un sistema UNIX de pequeña escala, desconectar la corriente no es un camino aceptable para dar de baja el sistema; no solamente podemos potencialmente perder datos y dejar el sistema de archivos en estado inconsistente, sino que también se pueden dañar ciertos tipos de controladores de disco que esperan tener un indicador de protección activado o que necesitan una orden para estacionar las cabezas después de que el sistema es apagado.

Sin embargo en el caso de suceder una inundación o a causa de fuego, es probablemente aconsejable desconectar la corriente si no se dispone del tiempo suficiente para dar de baja el sistema exitosamente. Los cuartos de cómputo al estilo antiguo frecuentemente tienen un botón de emergencia que apaga todo lo que este ahí de una sola vez.

3.5.3.2.2 Shutdown (la mejor manera de salir del sistema)

shutdown es el camino más seguro, más considerado y más correcto para reiniciar o dar de baja el sistema, o para regresar al modo usuario-único. Desafortunadamente casi todos los vendedores han decidido inventar sus propios argumentos pero las diferentes versiones de *shutdown* ofrecen resultados similares.

Se puede dar la orden *shutdown* y esperar mientras el sistema se da de baja. Durante el período de prueba, *shutdown* envía mensajes (con wall) a los usuarios que estén dentro del sistema durante periodos cortos antes de dar de baja el sistema. Por default, las advertencias siempre dicen que el sistema está dándose de baja y da el tiempo restante hasta el evento. También se puede definir un mensaje personalizado y sustituirlo por el que ofrece el sistema, dicho mensaje podría decir el porque el sistema esta siendo dado de baja y podría estimar el tiempo en que estará fuera antes de que los usuarios vuelvan a entrar nuevamente. En la mayoría de los sistemas, los usuarios no pueden entrar cuando el sistema se está dando de baja.

La mayoría de versiones de *shutdown* permiten especificar si el sistema va a salir completamente, va a reiniciar o se va a pasar a modo usuario-único; algunas veces se puede especificar también si se quiere hacer un chequeo con *fsck* después del arranque. En sistemas modernos con discos grandes, un chequeo con fsck puede tomar mucho tiempo; se puede normalmente evitar el chequeo si damos de baja limpiamente el sistema. En el tiempo apropiado, shutdown da de baja cuidadosamente el sistema y lo lleva al estado requerido.

3.5.3.2.3 halt: Una manera simple para dar de baja el sistema.

El comando *halt* hace el trabajo esencial para dar de baja el sistema. Es invocado por *shutdown –h,* pero puede ser además ser utilizado por sí mismo, *halt* registra el proceso de baja, elimina los procesos no-importantes, ejecuta la llamada al sistema *sync*, espera a que el sistema de archivos termine de guardar y escribir todo, y entonces detiene el kernel.

halt –n impide la llamada a *sync*; esto es utilizado después de reparar la partición root con fsck para prevenir que una reparación sobrescriba con versiones viejas del superbloque. *halt –q* provoca una salida inmediata sin sincronización, terminación de procesos o escritura de logs. Esta forma es rara vez utilizada.

3.5.3.2.4 Reboot: Una mala salida pero rápida.

reeboot es casi idéntico a *halt*, pero éste causa que la máquina reinicie nuevamente en vez de salir y apagar el sistema. reeboot es llamado por *shutdown –r*.

En adición a *halt* y *reboot*, algunos sistemas BSD proporcionan comandos llamados *fasthalt* y *fastboot*, estos comandos crean un archivo llamado */fastboot* antes de ejecutar *halt* o *reboot* respectivamente; la existencia de este archivo informa a los scripts de inicio para que mantengan los chequeos *fsck* haciendo el reinicio mucho más rápido.

3.5.3.2.5 Enviando una señal TERM a init.

Los resultados de terminar el proceso *init* son impredecibles y frecuentemente malos, se debe consultar la documentación antes de enviar cualquier señal. Cuando la versión BSD de *init* recibe una señal TERM, este normalmente destruye todos los procesos de los usuarios, daemons y otros procesos y regresa el sistema a modo usuario-único; esta facilidad es proporcionada por *shutdown*.

Para enviar una señal a un proceso, comúnmente necesitamos primero saber cual es la ID del proceso, utilizando *ps*; sin embargo, *init* es siempre el proceso número uno. Para enviar una señal, el comando *kill* es utilizado:

```
# sync
# kill –TERM 1
```

3.5.3.2.6 Cambiar el nivel de ejecución de init

En sistemas con un fuerte multi-nivel *init*, podemos utilizar el comando *telinit* para ir directo a un nivel de ejecución de *init*. Por ejemplo:

```
telinit s
```

Lleva al sistema a modo usuario-único sin dar el mencionado mensaje de advertencia que se obtiene con *shutdown*. El comando:

```
shutdown –is
```

Lleva al sistema al mismo estado de una mejor manera. *telinit* es el más útil para cambios y pruebas del archivo *inittab*. La opción *–q* hace que *init* vuelva a leer *inittab*.

3.5.3.2.7 Eliminando init

init es tan importante para la operación del sistema que si es eliminado, la mayoría de las computadoras reinician automáticamente (algunos kernels simplemente se paralizan). Esto es una manera muy ruda de reiniciar; es mejor utilizar *shutdown* o *reboot* en lugar de esto.

3.5.4 Usuarios root

Como en cualquier sistema operativo sano, el control administrativo de un sistema UNIX está separado del acceso general de los usuarios; el superusuario es un usuario poderoso que puede realizar tareas privilegiadas como el controlar procesos y adicionar dispositivos.

Los administradores de sistemas necesitan sobrepasar los mecanismos de protección en una gran variedad de situaciones, para hacer esto posible, UNIX procesa un UID diferente para todos los usuarios: UID cero para superusuario. Por convención, el sistema UNIX define una cuenta llamada "root" con este UID.

UNIX permite al superusuario realizar cualquier operación válida en un archivo o proceso. En adición a esto, algunas llamadas del sistema (requerimientos al kernel) pueden ser ejecutadas solo por el superusuario. Unas pocas llamadas al sistema están disponibles para todos los usuarios, pero tienen opciones especiales para root. Algunos ejemplos:

- Montar y desmontar sistemas de archivos.
- Cambiar el directorio root de un proceso con chroot.
- Crear archivos de dispositivo.
- Configurar el reloj del sistema.
- Cambiar la propiedad de los archivos (en sistemas BSD).
- Configuración del nombre del sistema.
- Configuración de las interfaces de red.
- Dar de baja el sistema.

Un ejemplo de los super-poderes del superusuario es la habilidad de adueñarse de un proceso cambiando la propiedad de este. El programa login es un caso de estos; el proceso que recoge el nombre de usuario y password cuando entramos a sesión inicialmente corre como root. Si el nombre de usuario y el password que se introducen son legítimos, login cambia sus UID y GID para el usuario especificado y ejecuta el shell para dicho usuario. Una vez que un proceso root cambia su propiedad para llevarla a un proceso normal de usuario, este no puede recuperar su forma de estado privilegiado.

3.5.4.1 Eligiendo un password para root

Es importante que el password de root sea seleccionado de tal manera que no pueda ser adivinado con facilidad o descubierto por un error; en teoría, el tipo más seguro de password consiste en una secuencia de letras, puntuaciones y dígitos aleatorios. Sin embargo este tipo de password es difícil de recordar y normalmente difícil de teclear, no suele ser óptimo y seguro si el administrador del sistema lo teclea de manera lenta.

Un password que consiste en dos palabras seleccionadas aleatoriamente separadas por signos de puntuación, o la primera letra de una frase favorita es siempre suficientemente segura. Los passwords de esta manera, además obedecen a algunos requerimientos de sistemas en el que todos los passwords deben tener una mezcla de al menos un número o caracter especial. El password de root debería ser de al menos ocho caracteres de longitud, tal que siete caracteres serían mucho menos seguro. No ayuda mucho el hecho de tener un password más grande de ocho caracteres, realmente solo los primeros ocho son significativos.

El password de root debería ser cambiado:

- Al menos una vez cada tres meses
- cada vez que alguien ajeno por alguna circunstancia conozca el password
- Cuantas veces se piense que la seguridad puede estar comprometida

3.5.4.2 Convertirse en superusuario

Hay muchas formas de accesar a la cuenta de superusuario. La más simple es entrar a sesión como root. Desafortunadamente salir de nuestra propia cuenta y entrar como root es frecuentemente inconveniente, una mejor manera de hacerlo es utilizar el comando *su*. Si su es invocado sin argumentos, pedirá solamente el password para root e iniciar el shell de root; los privilegios de este shell hacen efecto hasta que el shell termina.

El comando *su* es además capaz de reemplazar identidades diferentes de root. Si sabemos el algún password, podemos pasar directamente a esa cuenta simplemente ejecutando *su* seguido del nombre de usuario. Tal como con un su para root pedirá el password para el usuario especificado. En algunos sistemas, el password de root permite entrar a cualquier cuenta con *su*, en otros, debemos estar explícitamente como root antes de ir a otra cuenta.

Es buena idea tener el hábito de teclear */bin/su* en vez de simplemente *su*, esto da una mejor protección contra programas llamados su que se puedan haber sido encontrados por una ruta equivocada o puesto a propósito con la intención de capturar passwords.

3.5.4.1 Abusando del sistema

Junto con el sentimiento de poder, llega la tendencia a hacer daño, este es uno de los puntos oscuros de la administración de sistemas UNIX, pero todo el mundo pasa por ello en algún momento. Muchos usuarios de sistemas UNIX nunca tienen la posibilidad de manejar este poder en los sistemas UNIX de universidades y de empresas, solo los altamente cualificados (y altamente pagados) administradores de sistemas llegan a conectarse como root. De hecho, en muchas de esas instituciones, la clave de root es un secreto celosamente guardado. Es muy restringida la conexión como root; el poder es dado solo a un reducido grupo de elegidos.

Este tipo de actitud hacia la cuenta root es, sencillamente, el tipo de actitud que alimenta la malicia y el desprecio. Ya que root es tan atractivo cuando algunos usuarios tienen su primera oportunidad de conectarse como root, la tendencia es a utilizar los privilegios de root de forma descuidada. Yo he conocido "administradores de sistemas" (por llamarlos de

alguna forma) que leen el correo de otros usuarios, borran ficheros de usuario sin avisar y que, de forma general, se comportan como niños cuando se les da un "juguete" poderoso.

Puesto que root tiene tantos privilegios en el sistema, se necesita una cierta madurez y auto-control para utilizar la cuenta de la forma para la que esta diseñada, para administrar el sistema.

¿Cómo sentiríamos si el administrador de nuestro sistema UNIX leyese nuestro correo electrónico o mirase los archivos? Aun no hay ningún precedente legal firme acerca de la intimidad electrónica en sistemas de cómputo de tiempo compartido, en sistemas UNIX, el usuario root tiene la posibilidad de saltarse todos los mecanismos de seguridad y privacidad del sistema, es importante que el administrador del sistema desarrolle una relación de confianza con los usuarios del sistema; esto algo en lo que nunca se puede insistir lo suficiente.

3.5.5 Elementos para control de procesos

Ya explicamos anteriormente como se controla un proceso en el apartado "Administración de Procesos", sin embargo ahora analizaremos cuales son los componentes más importantes desde el punto de vista del administrador del sistema para el control de los procesos.

Un proceso es la abstracción de UNIX que maneja la memoria, CPU, y recursos de entrada/salida que comprende un programa ejecutándose. Aunque UNIX da la impresión de que muchas cosas se están ejecutando a la vez, solo un proceso se está ejecutando en un momento en particular. La ilusión de ejecución concurrente es guardada para una técnica llamada *división de tiempo*, en la cual UNIX cambia el proceso que se está ejecutando regularmente, intervalos cortos (normalmente cada 20 milisegundos). Los procesos se rotan rápidamente que parece que están cada uno corriendo al mismo tiempo, cuando en realidad cada proceso se está ejecutando solo en un pequeño porcentaje de tiempo.

El administrador del sistema tiene la habilidad para monitorear el estado de los procesos, controlar cuanto tiempo de CPU se le asigna a cada proceso, enviar indicaciones a un proceso, y suspender o terminar su ejecución.

3.5.5.1 Componentes de un proceso

Un proceso consiste en un espacio de dirección y un conjunto de estructuras de datos dentro del kernel. El espacio de dirección es un conjunto de páginas de memoria que el kernel ha marcado para ser utilizadas por los procesos; estas contienen segmentos para el código del programa que el proceso está ejecutando, las variables utilizadas por los procesos, las pilas utilizadas por los procesos y alguna información extra necesaria para el kernel mientras el proceso está corriendo.

Decimos que el espacio de dirección es una sección de memoria que actualmente es un espacio no real, porque en un sistema de memoria virtual el espacio de dirección puede ser todo, una parte, o no estar completamente en la memoria física en un momento dado.

La estructura interna de datos del kernel registra varias piezas de información acerca de cada proceso. Algunas de las más importantes son:

- El espacio de mapeo de la dirección del proceso.
- El estado actual de un proceso.
- La prioridad de ejecución de procesos.
- Información acerca de los recursos que los procesos han utilizado.
- La máscara de órdenes de procesos.
- Los dueños de los procesos.

Muchos de los parámetros asociados con un proceso directamente afectan su ejecución; un cierto tiempo del procesador es asignado, los archivos de este se pueden accesar etc. En seguida discutiremos el significado de los parámetros que son más interesantes desde el punto de vista del administrador.

PID

Cada nuevo proceso creado por el kernel es asignado a un número de identificación de proceso único. Como un número de seguridad social, el valor actual del PID tiene poca importancia. Los PIDs son asignados en el orden en que los procesos son creados; cuando el kernel sale del rango de PIDs, este comienza de nuevo desde cero, manteniendo aún los PIDs que están en uso.

PPID

UNIX no proporciona una llamada de sistema que cree un nuevo proceso ejecutando un programa particular, en lugar de eso, un proceso existente debe copiarse a sí mismo para crear un nuevo proceso; la copia puede intercambiar sus instrucciones para otro programa.

El proceso original es llamado "padre" del proceso, y la copia es llamada "hijo". El atributo PPID de un proceso es el PID de su padre.

UID y EUID

El UID es el número de identificación de la persona que ha creado el proceso. Solo el creador y el superusuario pueden hacer cambios al proceso.

El EUID es el UID "efectivo" del proceso. Este número es utilizado para determinar que recursos y archivos el proceso ha permitido accesar. Para la mayoría de los procesos, el UID y el EUID son lo mismo, la excepción la hacen los programas que forman el setuid.

GID y EGID

El GID es el número de identificación del grupo para los procesos, un número válido de grupos es enumerado en /etc/group, y en el campo GID del archivo /etc/passwd. Cuando un proceso es inicializado, su GID es dado por el GID de su padre.

El EGID está relacionado con el GID de la misma manera que el EUID está relacionado con el UID, tal que, si un proceso intenta accesar un archivo en el cual no tiene permiso del propietario, el kernel checa inmediatamente para revisar si los permisos han sido generados en términos de EGID.

En algunos sistemas, un proceso puede estar en más de un grupo al mismo tiempo, en este caso, el GID es actualmente una lista de números de grupos, y cuando se intenta accesar un recurso, la lista de entrada es checada para revisar si se pertenece al grupo apropiado.

3.5.5.2 Prioridad y mejor valor

La prioridad de un proceso determina que tanto tiempo del CPU se le asigna a este. Cuando el kernel selecciona un proceso para ejecutar, este selecciona el proceso con la más alta prioridad.

Es imposible establecer la prioridad interna directamente, pero si es posible establecer un "mejor valor" a un proceso, el cual tiene una substancial influencia en la prioridad interna. Otros factores que influyen en la prioridad interna son la cantidad de tiempo de CPU que el proceso ha consumido recientemente y la longitud de tiempo que éste ha esperado para ejecutarse nuevamente.

3.5.5.3 Terminal de control

La mayoría de los procesos tienen una terminal de control asignada a ellos. La terminal de control determina los vínculos internos para la entrada estándar, salida estándar y canales de error estándar. Cuando se inicia un comando desde el shell, la terminal normalmente obtiene la terminal de control del proceso. El concepto de una terminal de control además afecta la distribución de las señales.

3.5.5.4 El ciclo de vida de un proceso

Los procesos simplemente no aparecen de forma mágica en el sistema, no son creados espontáneamente por el kernel, los nuevos procesos son creados por otros procesos. Para crear un nuevo proceso, un proceso se copia a sí mismo utilizando la llamada al sistema fork; fork crea una copia del proceso original que es idéntico al padre excepto por las siguientes diferencias:

- El nuevo proceso tiene un PID diferente.
- El nuevo PPID del proceso hace referencia al proceso original.
- La información del registro del nuevo proceso se establece en blanco.
- El nuevo proceso tiene su propia copia de descriptores de archivos.

Los descriptores son números de referencia que el kernel da a los procesos cuando un archivo o socket es abierto; estos números son indexados dentro de una pequeña tabla que contiene punteros a la propia estructura de datos del kernel, cuando un fork ocurre, esta es la tabla que es copiada, no la estructura fundamental. La manipulación de las estructuras de los hijos puede tener un efecto directo en el padre.

Por ejemplo: supongamos que el proceso hijo lee algunos datos del descriptor de archivo, la siguiente vez que el padre intente leer del descriptor, éste comenzará leyendo en el lugar donde el hijo dejó de hacerlo, no desde donde debería haber comenzado antes del fork.

fork tiene la única propiedad de regresar dos valores. Desde el punto de vista de los hijos, fork siempre regresa cero; con el padre por otro lado, devuelve el PID del nuevo hijo creado, así es como los dos procesos se distinguen. En C esto se vería como sigue:

```
int kidpid;
kidpid = fork();
if (kidpid == 0) {
 /* Aquí está el proceso hijo*/
} else {
 /* aquí está el padre*/
}
```

Después de un fork, un nuevo proceso utilizará una de las llamadas al sistema para iniciar la ejecución del programa.

3.5.5.5 Señales

Una señal es una forma de decir a un proceso "algunas cosas interesantes para hacer, tal como: ¡deja de hacer lo que estas haciendo y haz esto!". Más de treinta tipos de señales están definidas. Las señales pueden ser enviadas a un proceso a través del comando *kill*.

Cuando una señal es enviada, una de dos cosas puede suceder: si el proceso ha diseñado una rutina particular para cada señal, esta rutina es llamada con información acerca del contexto con el cual la señal fue enviada, de otra manera, el kernel toma alguna acción por omisión en representación del proceso. La acción por omisión varía de señal a señal; muchas señales terminan el proceso, y algunas generan un archivo de error llamado "core".

Especificar un manejador de rutina para una señal es referirse a atrapar la señal. Cuando un manejador termina, la ejecución se reestablece en el punto en donde la señal fue recibida.

Para prevenir señales entrantes, los programas pueden requerir que estas sean ignoradas o bloqueadas, una señal que es ignorada es simplemente descartada y no tiene efecto en el proceso, una señal bloqueada se va a la cola de entrega, pero el kernel no requiere que el proceso actúe, hasta que ésta sea explícitamente desbloqueada. El manejador para una nueva señal desbloqueada es llamado solo una vez, aún, si múltiples casos de señales fueron recibidos mientras la señal fue bloqueada.

Las señales son utilizadas en muchas formas. Las señales pueden ser enviadas en medio de los procesos como una forma de comunicación a través del kernel en cualquier situación, o a través del proceso a sí mismo.

Un simple ejemplo de acción de señal consideremos que sucede cuando los caracteres Control-C son tecleados; el controlador de terminal recibe el caracter y envía una señal INT (interrupción) para el grupo de proceso activo, desde ahí, el manejador por omisión para INT especifica la terminación. La mayoría de los programas aborta cuando reciben esta señal.

3.5.6 Dispositivos y Controladores

Un controlador de dispositivo es un programa que maneja la interacción del sistema con una pieza particular de hardware. El controlador traduce entre el lenguaje del hardware y la interfaz de programación utilizada por el kernel. La existencia de un controlador ayuda a mantener en UNIX una cierta independencia de dispositivos.

Los controladores de dispositivos son parte del kernel; estos no son procesos de usuario, sin embargo, un controlador puede ser accesado en ambos, desde dentro del kernel y desde el espacio del usuario. El acceso a nivel de usuario es proporcionado a través de archivos de dispositivo especiales que se encuentran en el directorio /dev; el kernel transforma las operaciones en esos archivos especiales en llamadas para el código del controlador.

En los años 80's, la mayoría de las piezas requeridas, tarjetas de interfaz y controladores de dispositivos en las tarjetas madre eran compilados dentro del kernel. Muchas cosas han pasado para cambiar esto:

1. La adopción del estándar SCSI para la mayoría de los controladores de discos y cintas han eliminado la necesidad para los fabricantes de crear sus propias especificaciones de interfaces.

2. La implementación de módulos cargables del kernel ha permitido adicionar controladores al kernel sin reconectar.

3. Los fabricantes de estaciones de trabajo ahora incluyen muchos dispositivos a ser utilizados opcionalmente en sus configuraciones estándar.

4. El mercado de UNIX ha sido grande y sigue creciendo. Los vendedores de hardware han descubierto que no pueden vender hardware que no es soportado.

3.5.6.1 Categorías de Hardware

La mayoría de los dispositivos que se adicionan a un sistema UNIX, pueden caer dentro de tres categorías:

SCSI

Los dispositivos que se adicionan a un bus SCSI son fáciles de configurar. La mayoría de los sistemas tienen un controlador que permite accesar directamente al bus SCSI, además de controladores adicionales para tipos especiales de dispositivos como los discos y cintas.

Proveedor

La mayoría de los dispositivos que se pueden comprar a un proveedor de hardware pueden actualmente ser soportados por los sistemas operativos UNIX. A menos que se tenga un controlador muy exclusivo, el kernel reconocerá el nuevo dispositivo al momento de instalar dicho controlador. En adición, el archivo de dispositivo en /dev debe ser creado por el usuario, o el sistema creará este durante el proceso.

Algunos vendedores (por ejemplo, Sun) publican a través de documentaciones el hardware "third-party" que se puede configurar por si mismo tal como lo haría el vendedor. (Algo similar a lo que Microsoft llama "plug and play"). Así que probablemente no se tenga que hacer ninguna configuración cuando se adicionan tarjetas ethernet, tarjetas gráficas y controladores SCSI.

Third-party

Cuando se compra una pieza de hardware "third-party", alguien diferente al proveedor de hardware, este normalmente se adiciona con un script que instala el dispositivo, reconfigura el kernel (si es necesario), y crea entradas de dispositivo. Muy pocas instrucciones paso por paso son proporcionadas. En sistemas con controladores cargables, la reconfiguración del kernel no es necesaria.

3.5.6.2 Números de Dispositivo y Tablas de Relación

Los archivos de dispositivo son mapeados hacia dispositivos a través de "el mayor y menor número de dispositivo", los valores son almacenados en la estructura del inodo del archivo. El mayor número de dispositivo identifica el controlador al que el archivo está asociado. El menor número de dispositivo identifica cual dispositivo en particular de un tipo dado es direccionado. El menor número de dispositivo es frecuentemente llamado el "número de unidad" o "instancia" del dispositivo.

El menor número de dispositivo es algunas veces utilizado por el controlador para seleccionar características particulares de un dispositivo, por ejemplo, un simple controlador de cinta, puede tener muchos archivos de representación en /dev con varias configuraciones de densidad de grabación y características de rebobinado.

Hay actualmente dos tipos de archivos de dispositivo: archivos de dispositivo de bloque y archivos de dispositivo de caracter. Un dispositivo de bloque es leído o escrito por bloques (un grupo de bytes, normalmente un múltiplo de 512) cada vez, mientras que los dispositivos de caracter pueden ser leídos o escritos un byte cada vez. Algunos dispositivos soportan accesos a través de ambos, bloques y carácter.

Los controladores de dispositivo presentan una interfaz estándar al kernel. Cada controlador tiene rutinas para desempeñar todas o algunas de las siguientes rutinas:

```
probe     attach    open close     read reset     stop
select    strategy  dump psize     write          timeout
```

```
process a transmit interrupt
process a receive interrupt
ioctl (input/output control)
```

Dentro del kernel, la dirección de estas funciones para cada controlador son almacenadas en una estructura llamada "tabla de relación". Hay actualmente dos tablas: una para dispositivos caracter y una para dispositivos de bloque. La tabla de relación es indexada por el mayor número de dispositivo.

Cuando un programa realiza una operación en un archivo de dispositivo, el kernel automáticamente atrapa la referencia, busca el nombre de la función apropiada en la tabla de referencia y transfiere el control a ésta. Para realizar una operación inusual que no tiene una representación directa en el modelo del sistema de archivos (por ejemplo, sacar un disco flexible), la llamada al sistema *ioctl* es utilizada para pasar un mensaje directamente del espacio del usuario dentro del controlador.

Todos los sistemas UNIX proporcionan alguna manera para adicionar entradas a las tablas de referencia. En los viejos sistemas UNIX donde en código fuente era proporcionado, simplemente se editaba el código en C y se hacían los cambios. La mayoría de los proveedores tiene ahora desarrollado un sistema para construir una tabla de referencia desde uno o más archivos de texto; hay normalmente un archivo adicional o conjunto de archivos que determinan cuales módulos serán enlazados dentro del kernel.

3.5.6.3 Archivos de dispositivo

Por convención, los archivos de dispositivo son almacenados en el directorio /dev. En grandes sistemas, especialmente aquellos conectados en red y pseudo terminales, puede haber cientos de dispositivos. Los sistemas basados en AT&T manejan esto de una manera muy sencilla utilizando un subdirectorio separado de /dev para cada tipo de dispositivo: discos, cintas, terminales, etc.

Los dispositivos son creados con el comando mkmod, el cual tiene la siguiente sintaxis:

mkmod *nombre_archivo tipo mayor menor*

Dónde nombre_archivo es el dispositivo a ser creado, tipo es "c" para dispositivos caracter y "b" para dispositivos bloque, mayor y menor son el mayor y el menor número de dispositivo. Si se está creando un archivo de dispositivo que se refiere a un dispositivo que está actualmente presente en el kernel, hay que checar las páginas del manual para el controlador y encontrar el apropiado número mayor y menor para el dispositivo.

Un script de shell llamado MAKEDEV es algunas veces proporcionado en /dev para automáticamente sustituir los valores por omisión para mknod. Debemos buscar a través del script para encontrar los argumentos necesarios para el dispositivo. Por ejemplo, para crear entrada PTY en un sistema SunOs, pudiéramos utilizar el siguiente comando:

```
# cd /dev
# ./MAKEDEV pty
```

3.5.6.4 Módulos cargables del kernel

La mayoría de nuestros sistemas de referencia soportan módulos cargables del kernel. Los módulos cargables permiten que los controladores de dispositivo sean enlazados y removidos del kernel mientras se está ejecutando, esto hace que la instalación de los controladores sea mucho más fácil, puesto que el kernel binario no tiene que ser cambiado. Esto además permite un kernel más pequeño porque los controladores no son cargados a menos que sean necesarios.

Los módulos cargables son implementados porque se proporcionan uno o más documentos "hooks" dentro del kernel donde controladores de dispositivo adicionales pueden cargarse. Un comando a nivel de usuario se comunica con el kernel y le dice a este que cargue nuevos módulos dentro de la memoria y crea entradas para ellos en la tabla de relación del sistema. Hay normalmente un comando a nivel de usuario que descarga los controladores.

A pesar de que los controladores cargables son convenientes, estos no son completamente seguros, en cualquier momento en que se carga o descarga un módulo, se corre el riesgo de causar un conflicto en el kernel. No se recomienda cargar y descargar un módulo no probado cuando no se está dispuesto a arriesgar y abortar el sistema.

Como otros aspectos de manejo de dispositivos y controladores, la implementación de módulos cargables es dependiente del sistema operativo. Ejemplos de sistemas que soportan módulos cargables son: Solaris, HP-UX, IRIX y SunOS.

3.5.7 Configuración del kernel

Un sistema UNIX tiene tres capas esenciales:

- El hardware.
- El kernel del sistema operativo.
- Los programas de usuario.

El kernel oculta el hardware del sistema dentro de una abstracción, una interfaz de alto nivel de programación; esta es responsable de la implementación de muchas facilidades que usuarios y programas de usuario utilizan, por ejemplo, el kernel ensambla todos los siguientes conceptos desde las características de bajo nivel del hardware.

- Procesos (tiempo compartido, espacios de dirección).
- Señales y semáforos. (interrupciones).
- Memoria virtual.(swapping, paging, mapping).
- El sistema de archivos (archivos, directorios).
- Tuberías y conexiones de red (comunicación entre procesos)

El kernel contiene controladores de dispositivo los cuales manejan piezas determinadas de hardware; el resto del kernel es como un gran dispositivo independiente. La relación entre el kernel y sus controladores de dispositivo es similar a la relación entre los procesos a nivel de usuario y el kernel. Cuando un proceso pide al kernel "leer los primeros 64 bytes de /etc/passwd", el kernel podría trasladar éste dentro de una instrucción del controlador de dispositivo como "traer el bloque 3,348 desde el dispositivo 3". El controlador ejecutará este comando en secuencias de patrones de bit presentados en el registro de control del controlador.

El kernel está escrito la mayor parte en C, con un poco de lenguaje ensamblador para los procesos de más bajo nivel. Muchos años atrás, un kernel compilado de UNIX era considerablemente modesto en tamaño, comúnmente medía solo medio megabyte; hoy, con el trabajo de las redes, los sistemas de archivos de red, multi-hilos, y la memoria barata, los kernels miden alrededor de 700Kb a 2 Mb.

3.5.7.1 Diferencias entre sistema V (AT&T) y BSD

Es esencial que el kernel esté alerta y comunicado con el hardware. En ambos, BSD y AT&T, se puede proporcionar el kernel con información especifica del hardware que espera encontrar o no encontrar.

Es un mito popular el que el kernel de los sistemas AT&T se configuran a sí mismos mientras que los BSD no lo hacen. Algo más atinado es que los kernels de sistemas BSD dan una guía durante el avance, mientras que los de AT&T no lo hacen y el proceso se lleva a cabo ciegamente, algunas veces causan errores fatales.

Esto filosóficamente se divide y se enfoca a diferencias substanciales en el proceso de configuración. Configurar y construir un kernel es un rito pasajero para un administrador hoy en la actualidad, porque existen muchas mejoras para realizar este proceso; esto normalmente involucra algunos parámetros operacionales al kernel y le informa acerca del hardware que se está utilizando.

3.5.7.2 Cuando configurar el kernel

Hay pocas situaciones en las cuales es necesaria la reconfiguración del kernel. En esta sección describimos esto.

Instalando un nuevo sistema

Se debería reconfigurar el kernel cada vez que se actualice la máquina y cada vez que se actualice el sistema operativo. La mayoría de las versiones de UNIX llegan con un kernel "genérico" configurado. Deberían normalmente ser proporcionados con ambas configuraciones, un archivo de configuración genérica y otro correspondiente al kernel ejecutable.

El kernel genérico probablemente trabaja bien, pero se debería reconfigurar por razones de eficiencia. Los kernels genéricos son diseñados para estar con un tipo de hardware que se

integra con cuidado al sistema; estos frecuentemente incluyen todo lo que es comúnmente utilizado por los controladores de dispositivo, todos los pseudo dispositivos, y la mayoría de las opciones del kernel. Esto los hace flexibles, pero muy pesados, la reconfiguración inicial en la mayoría de las veces es un proceso de eliminación de controladores para dispositivo que no se tienen y hay que eliminar opciones que no se utilizan.

3.5.7.3 Agregar controladores de dispositivo

Para adicionar un nuevo tipo de dispositivo al sistema, se debe incluir su controlador al kernel. El código controlador debe ser integrado dentro del las estructuras de datos del kernel y tablas.

En algunos sistemas, eso puede requerir ir a configurar archivos para el kernel y agregar los nuevos dispositivos, reconstruyendo el kernel para su correcto funcionamiento. En otros sistemas, esto puede requerir la ejecución de un programa diseñado para hacer dichos cambios en la configuración.

Algunos sistemas incluyen el concepto de un controlador de dispositivo "agregable", en la mayoría de los casos implica que un nuevo código puede ser cargado dentro del kernel mientras está ejecutándose.

3.5.8 Seguridad del sistema

Debe tenerse en presente que la seguridad es relativa. La seguridad de cada sistema se implementa de una manera diferente de acuerdo a las necesidades, por lo tanto existen diferentes grados de seguridad; cada uno de ellos proporciona una protección limitada y pueden ser evitadas por usuarios con conocimientos, pero muchos pueden ser atacados con éxito por expertos.

Proporcionar seguridad que sea altamente resistente procedimientos y técnicas especiales es un trabajo dificil. Tales características pueden hallarse en versiones especiales del sistema UNIX desarrolladas para conseguir requerimientos de seguridad demandados por el departamento de defensa de los Estado Unidos como el UNIX System V/MLS.

3.5.8.1 Seguridad en base a contraseñas

La primera línea de defensa contra acceso no autorizado a un sistema es la protección mediante contraseñas; es también a menudo el eslabón más débil de la cadena porque la realidad es que los usuarios solo desean utilizar contraseñas sencillas y fáciles de recordar, no desean cambiarlas y les gusta escribirlas para poder consultarlas. Desgraciadamente para el administrador del sistema, estas son malas costumbres desde el punto de vista de la seguridad informática. La seguridad basada en contraseñas requiere de una atención casi constante.

La contraseña de root ya la tratamos en un apartado atrás, cualquiera que la conozca puede accesar a cualquier parte del sistema y tal vez estar conectado a otros a través de una red. Modificar las contraseñas muy a menudo es una buena práctica, es mejor memorizarla que

escribirla en algún lugar, en la mayor parte de las organizaciones es una buena práctica que dos personas la conozcan, pero no más de dos.

Para seleccionar una buena contraseña, una técnica consiste en seleccionar dos palabras cortas aleatoriamente y conectarlas con un carácter de puntuación. Ésta combinación da como resultado una secuencia casi aleatoria de caracteres para el "cracker" pero es bastante fácil de recordar para el usuario. A continuación se proporcionan algunos ejemplos de contraseña que utilizan ésta técnica:

juan&dia
car!pan
modem!uno

Otro método para seleccionar contraseñas es tomar una frase que se recuerde fácilmente y utilizar la primera letra de cada palabra. El resultado es una secuencia aleatoria de caracteres que se puede recordar con facilidad, por ejemplo, la frase "Señoras y Caballeros, esto se acabó" se convertirá en la contraseña:

S&Cesa

La cuestión vital es que la contraseña debe memorizarse y no ser anotada en ningún sitio.

3.5.8.2 Seguridad de conexión al sistema

Cada cuenta del sistema UNIX es una puerta para entrar al servidor, todo lo que necesita alguien para entrar es la contraseña correcta. Si ya se han instituido buenas prácticas de gestión de contraseñas, ya se cuenta con una gran ventaja en lo que respecta a un sistema más seguro. Otro aspecto de la seguridad informática que acompaña a las contraseñas es la de seguridad de conexión al sistema.

La seguridad de conexión al sistema implica la búsqueda de cuentas en el sistema que puedan constituir problemas de seguridad potenciales, así como tratar con ellas. Hay varios tipos de problemas que pueden derivarse de ellas.

3.5.8.2.1 Cuentas sin contraseña

Muchos "crackers" informáticos penetran con éxito en los sistemas informáticos sencillamente encontrando una cuenta que no tenga una contraseña; se debe comprobar continuamente el archivo de contraseñas para encontrar estas cuentas y desactivarlas. Las contraseñas se almacenan en el segundo campo del archivo de contraseñas. Se puede comprobar la existencia de un campo de contraseña en blanco utilizando varias herramientas como por ejemplo, grep, awk, perl. Se puede desactivar la conexión a una cuenta editando al archivo de contraseñas y modificando el campo de contraseñas con un carácter "*", esto impide que alguien pueda conectarse al sistema utilizando ese nombre de usuario.

3.5.8.1.2 Cuentas no utilizadas

Si no se piensa volver a utilizar un nombre de usuario para conexión al sistema, se debe borrar la cuenta de forma que no se pueda utilizar ilegalmente. Podemos utilizar el comando userdel para borrar el nombre de usuario. Por ejemplo, para borrar al usuario alex39 se utiliza:

```
userdel alex39
```

Esto borra el nombre de conexión alex39 e impide que nadie pueda utilizarlo para conectarse al sistema. Se utiliza la opción *–r* de *userdel* si se desea borrar los archivos que pudiera haber en el directorio de usuario alex39.

3.5.8.2.3 Cuentas predeterminadas

Algunas versiones de UNIX vienen con varios identificadores estándar necesarios para algunos servicios. Por ejemplo, la cuenta de root no tiene contraseña alguna cuando UNIX se instala por primera vez. Se debe comprobar el archivo de contraseñas una vez que se ha finalizado la instalación a fin de asegurarse de que todas las cuentas predeterminadas tienen contraseñas válidas o que se han desactivado escribiendo un caracter de asterisco "*" en el campo correspondiente a la contraseña.

Algunos paquetes de software crean cuentas automáticamente en el sistema durante sus procesos de instalación. No debemos olvidar desactivar o establecer contraseñas, según se requiera.

3.5.8.2.4 Cuentas de invitados

No es extraño que los centros informáticos dispongan de algún tipo de cuenta de acceso como invitado para visitantes, a fin de que estos puedan utilizar los ordenadores locales temporalmente. Estas cuentas normalmente no tienen contraseñas o tienen contraseñas que son iguales a los identificadores de usuario. Por ejemplo, el acceso a conexión *guest* podría no tener contraseña establecida o utilizar la contraseña *guest*. Como se puede adivinar, los atentados contra la seguridad están ahí esperando una oportunidad.

Debido a que estas cuentas y contraseñas son por lo general conocidas, un intruso podría utilizar una para acceder inicialmente al sistema, una vez que un "cracker" ha penetrado al sistema, el intruso podrá intentar conseguir acceso raíz desde el interior o utilizar el sistema como *punto de acceso* para atacar otros ordenadores en la red; el seguimiento de uno de estos ataques a una cuenta abierta y pública hace que sea mucho más difícil encontrar la fuente real del ataque.

Las cuentas de invitado abiertas no son aconsejables en ningún sistema. Si realmente se necesita una, debe mantenerse desactivada hasta que se necesite.

3.5.8.2.5 Cuentas de acceso de comando

Es normal que los ordenadores tengan varias cuentas de acceso de comando (identificadores de conexión al sistema que ejecutan un comando determinado y salen). Por ejemplo, *finger* es una cuenta que no tiene contraseña alguna; cuando un usuario se conecta al sistema como *finger*, se ejecuta el programa *finger*, mostrando quien está en el sistema y entonces la sesión finaliza. Otras cuentas parecidas podrían ser *sync* o *date*, estas cuentas normalmente no tienen contraseñas. Aunque no ejecutan órdenes sino tan solo comandos, estas cuentas también pueden constituir un riesgo contra la seguridad. Si se permite la existencia de cuentas de comando en el sistema, debemos cerciorarnos de que ninguno de estos comandos acepta entrada de línea de comandos. Además, estos comandos no deberán tener ningún tipo de escape shell que pueda permitir a un usuario acceder a un shell interactivo.

Una segunda razón para no utilizar estos tipos de cuenta es que proporcionan información sobre el sistema que puede ser de utilidad a un intruso. La utilización de programas como, por ejemplo *finger* o *who* como cuentas de comando pueden permitir a los intrusos obtener identificadores de usuario para realizar la conexión al sistema. Si un intruso llega a conseguir el identificador de conexión de un usuario, habrá conseguido ya la mitad de la información que necesita para conectarse a esa cuenta.

3.5.8.2.6 Cuentas de grupo

Una cuenta de grupo es una cuenta para la que más de una persona conoce la contraseña y donde estas personas se conectan bajo el mismo identificador. Es una mala idea, si se tiene una cuenta compartida por varias personas, y alguien penetra y la utiliza como base para atacar otros sistemas, en este caso será difícil encontrar a la persona que ha proporcionado la contraseña. Si se tiene una cuenta compartida por 5 personas, puede ser que en realidad este siendo compartida por 25 o más. No hay forma de saberlo con seguridad.

UNIX proporciona la capacidad de permitir un acceso a archivos basado en la pertenencia a un grupo. De esta forma, un grupo de personas que necesite acceso a un juego de archivos podrá compartirlo sin necesidad de compartir una cuenta.

3.6 ADMINISTRACIÓN DE UNA RED TCP/IP EN UNIX

El conjunto de protocolos ampliamente utilizados y conocidos como "Transmission Control Protocol / Internet Protocol" (TCP/IP) es cada vez más importante ya que de él dependen importantes redes nacionales e internacionales como Internet.

TCP/IP surgió inicialmente como un proyecto promovido por el gobierno de los E.U.A hasta alcanzar su extenso uso actual conectando redes de todos los tamaños. Reconocido por su capacidad para permitir comunicaciones entre diferentes plataformas de cómputo, se encuentra virtualmente en todas las estaciones de trabajo, minicomputadoras y computadoras de gran tamaño.

3.6.1 Comprensión del conjunto de protocolos TCP/IP

A mediados de los años 70, el Departamento de Defensa de los Estados Unidos (DOD) reconoció el desarrollo de un problema de comunicaciones electrónicas dentro de la organización. La comunicación del cada vez mayor volumen de información electrónica entre el personal de DOD, laboratorios de investigación, universidades y contratistas se enfrentaban un importante obstáculo. Las diferentes entidades tenían sistemas informáticos procedentes de diferentes fabricantes que ejecutaban sistemas operativos diferentes y utilizaban diferentes topologías y protocolos de red.
Se pidió a la agencia de investigación y de proyectos avanzados (ARPA) que resolviera el problema que suponía tratar con diferentes equipos y topologías de red. ARPA formó una alianza con universidades y fabricantes de sistemas informáticos para el desarrolló de estándares de comunicación [Trackett, 1998]. Esta alianza especificó y desarrollo una red de cuatro nodos, que es la base de la red Internet actual. Durante los años 70 esta red emigró a un nuevo diseño de protocolo central que se convirtió en la base de TCP/IP [Trackett, 1998].

La mención de TCP/IP requiere de una breve introducción a Internet. La red Internet conecta a cientos de miles de computadoras. Sus nodos incluyen universidades, importantes empresas y laboratorios de investigación en los muchos países. Es un repositorio para millones de programas de uso compartido, noticias sobre cualquier tema, foros públicos, intercambio de información y correo electrónico. Otra característica es la conexión remota a cualquier sistema informático de la red utilizando el protocolo telnet. Debido al número de sistemas que están interconectados, pueden compartirse masivos recursos informáticos, permitiendo así la ejecución de grandes programas en sistemas remotos.

3.6.1.1 Análisis de la pila de protocolos TCP/IP

La pila de protocolos TCP/IP representa una arquitectura de red similar al modelo de red OSI de ISO. La figura siguiente muestra el mapa de las capas TCP/IP en la pila de protocolos ISO.

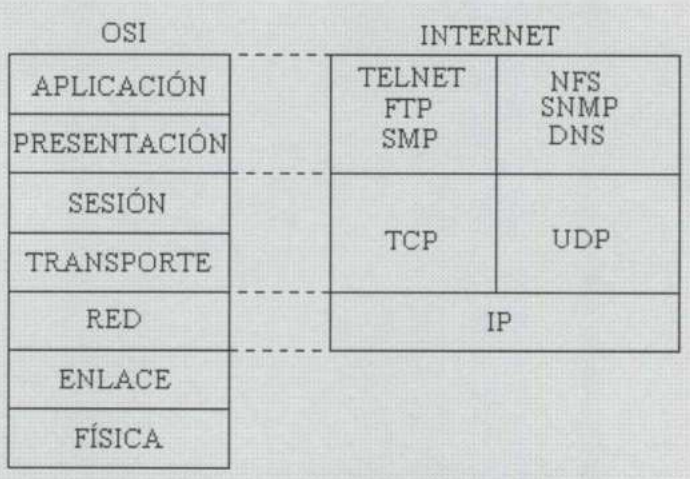

Figura 3-4 Comparación de OSI y TCP/IP

TCP/IP no establece tantas distinciones como OSI entre las capas superiores de la pila de protocolos. Las tres capas superiores OSI vienen a ser el equivalente de los protocolos de proceso de Internet. Algunos ejemplos de protocolos de proceso son Telnet, FTP, SMNP, NFS, SNMP y DNS.

La capa de transporte del modelo OSI es responsable de la entrega fiable de datos. En la pila de protocolos de Internet, esto corresponde a los protocolos sistema principal a sistema principal. Los ejemplos de esto son TCP y UDP. TCP se utiliza para traducir mensajes de longitud variable procedentes de los protocolos de la capa superior y proporciona el necesario acuse de recibo y control de flujo orientado a conexiones entre sistemas.

UDP es similar a TCP salvo que no está orientado a conexiones y no acusa recibo de los datos. UDP sólo recibe mensajes y los transmite a los protocolos de nivel superior. UDP proporciona una interfaz mucho más eficaz para acciones como servicios remotos de disco, debido a que no admite las cargas relacionadas con TCP.

El protocolo de Internet IP es responsable de comunicaciones sin conexiones entre sistemas. Se asigna en el modelo OSI como parte de la capa de red. La capa de red del modelo OSI es responsable del movimiento de información en la red. Esto se consigue examinando la dirección de la capa Red. Esta dirección determina los sistemas y la ruta a utilizar para enviar el mensaje.

IP proporciona la misma funcionalidad que la capa de red y ayuda a enviar mensajes entre sistemas, pero no garantiza la entrega de estos mensajes. IP puede también fragmentar los mensajes en pedazos y volver a unirlos en su destino. Cada uno de los fragmentos puede tomar una ruta diferente de red entre sistemas. Si el fragmento llega fuera de orden, IP reconstruye los paquetes en su secuencia correcta en el destino.

3.6.1.2 Direcciones IP

El protocolo de Internet requiere que se asigne una dirección a cada uno de los dispositivos de red. Esta dirección es conocida como la dirección IP y está organizada como una serie de cuatro octetos. Cada uno de estos octetos define una dirección única, en la que parte de la dirección representa una red (y opcionalmente una subred) y la otra parte representa un nodo especifico de la red.

Algunas direcciones tienen significados especiales en Internet como se describe a continuación:

- Una dirección que empiece con un cero hace referencia al nodo local dentro de la red actual. Por ejemplo, 0.0.0.23 hace referencia a la estación de trabajo 23 en la red actual. La dirección 0.0.0.0 hace referencia a la estación de trabajo actual.
- La dirección de bucle interno 127 es importante en el proceso de resolución de problemas y diagnosis de red. La dirección de red 127.0.0.0 es el bucle interno local dentro de una estación de trabajo.
- La dirección ALL es representada por la activación de todos los bits, proporcionando un valor de 255. Por lo tanto, 192.18.255.255 envía un mensaje a todos los nodos de la red

192.18; de la misma forma 255.255.255.255 envía un mensaje a cada nodo de Internet. Es importante utilizar estas direcciones para mensajes de transmisión múltiple y avisos de servicio.

Es importante no utilizar 0, 127 o 255 al asignar números de nodo a las estaciones de trabajo, ya que estos números están reservados y tienen significados especiales.

3.6.1.3 Clases de direcciones IP

Las direcciones IP se asignan en rangos llamados clases, dependiendo de la aplicación y del tamaño de la organización. Las clases más comunes son A, B y C. Estas tres clases representan el número de bits localmente asignables disponibles para toda la red. La tabla 3-21 muestra las relaciones entre las diferentes clases de direcciones.

Clase	Nodos disponibles	Bits iniciales	Dirección de comienzo
A	$2^{24} = 1677772$	0xxx	0-127
B	$2^{16} = 65536$	10xx	128-191
C	$2^{8} = 256$	110x	192-223
D		1110	224-239
E		1111	240-255

Tabla 3-21 Clases de direcciones IP

3.6.2 Configuración de una red TCP/IP

La configuración de una red TCP/IP es una de las tareas más comunes con las que se encuentra un administrador UNIX. En la mayoría de los casos básicos no es muy compleja, sin embargo, requiere conocimientos mínimos del diseño de la red y algunos programas y archivos de configuración.

3.6.2.1 Archivo de configuración TCP/IP

La conexión de red TCP/IP de UNIX está controlada por un conjunto de archivos de configuración en el directorio /etc. Estos archivos informan a UNIX de su dirección IP, nombre del sistema y nombre de dominio, y controlan las interfaces de red. La tabla 3-22 muestra qué hace cada archivo.

Archivo	Descripción
/etc/hosts	Asigna los nombres de sistema a las direcciones IP.
/etc/networks	Asigna los nombres de dominio a las direcciones de la red.
/etc/rc* o /etc/init.d	Secuencia que configura y activa las interfaces ethernet en el momento del arranque.

Tabla 3-22 Archivos de configuración de la red TCP/IP de UNIX

Las direcciones de clase A se utilizan para redes de gran tamaño o para colecciones de redes asociadas. Casi todas las instituciones educativas están agrupadas bajo una dirección de clase A. Las direcciones de clase B se utilizan para redes de gran tamaño con más de 256 nodos (pero con menos de 65,536 nodos). Las direcciones de clase C son utilizadas por casi todas las organizaciones. Es aconsejable que una organización obtenga varias direcciones de clase C debido a que el número de direcciones de clase B está limitado. La clase D está reservada para los mensajes de transmisión múltiple de la red, mientras que la clase E está reservada para experimentos en desarrollo.

3.6.2.1 El archivo /etc/hosts

Toda computadora dentro de una red TCP/IP tiene una dirección IP, un nombre de sistema canónico y opcionalmente uno a más alias de nombre de sistema. El archivo /etc/hosts es el método original para asignar nombres de sistema a direcciones IP. A modo de ejemplo, se examinará la red que construyó Tristar Inc., esta red se compone de una única dirección de red de Clase B asignada a Tristar por el NIC [Trackett, 1998]; esta red se ha dividido en dos subredes de clase C. El formato del archivo de sistemas se muestra en el ejemplo siguiente:

```
# /etc/hosts for unix.tristar.com
#
#For loopbacking.
127.0.0.1               localhost

#This machine
166.82.1.21             unix.tristar.           #the local
                        com unix1               machine

#Other hosts on our netwrok

166.82.1.20             server.tristar.         #the server
                        com server

166.82.1.22             wk1.tristar.com         #workstation 1
166.82.10               netpr1.tristar.         #networked
                        com netpr1              printer

166.82.1.1              gateway.tristar.        #the router
                        com gateway

166.82.1.1              gate-if1                #1st Interface
                                                on gateway

166.82.2.1              gate-if2                #2nd interface
                                                on gateway

166.82.1.30             unixlt.tristar.         #Laptop via PLIP
                        com unixlt

#end of hosts file
```

El formato del archivo hosts se compone de una dirección IP por línea, comenzando en la primera columna, el nombre de sistema canónico asociado con esta dirección y opcionalmente uno o más alias. Los campos están separados por espacios o tabuladores. Las líneas en blanco y el texto que sigue a un carácter # se tratan como comentarios y se ignoran.

La dirección IP 127.0.0.1 se conoce como la dirección de bucle interno local y se reserva para este propósito. Generalmente se le asigna el nombre *localhost*. Si se va a utilizar el sistema solo, como un sistema autónomo, sólo se necesita la dirección *localhost* en el archivo de sistema.

La función del archivo /etc/hosts ha sido asumida en gran parte por el Servicio de Nombres de Dominio (DNS) en los sistemas conectados Internet o a grandes redes internas. Sin embargo, DNS no está disponible durante el arranque o cuando se está trabajando en el modo de usuario-único, por lo que es buena idea colocar la información para los sistemas esenciales como los servidores y ruteadores en /etc/hosts.

En una red con pocos sistemas que no este conectada a Internet, es más fácil mantener una lista completa de todos los sistemas en /etc/hosts en lugar de configurar y mantener el DNS.

3.6.2.2 El archivo /etc/networks

Al igual que los sistemas que pueden tener nombres y direcciones IP, las redes y subredes también pueden denominarse. Esta denominación la maneja el archivo /etc/networks. El siguiente es un archivo de ejemplo para tristar.com [Trackett, 1998].

```
# /etc/networks for tristar.com

Localnet        127.0.0.0       #software loopback
                                network

tristar-c1      166.82.1        #Development group
                                Network, Class C

tristar-c2      166.82.2        #MISNetwork, Class C

#end of network file
```

El primero es el nombre de localnet y la dirección IP 127.0.0.0. Si no se conecta el sistema UNIX a una red TCP/IP, esto es todo lo que se necesita poner en el archivo. Las líneas siguientes identifican las dos subredes de Clase C que Tristar ha hecho a partir de su red de Clase B.

Las direcciones IP del archivo de redes incluyen sólo la parte de dirección de la red, más el byte de la subred.

3.6.3 Configuración de Servicio de Nombres de Dominio (DNS)

DNS suministra un mecanismo para la conversión de direcciones IP a nombres mnemónicos que representan sistemas, redes y alias de correos. Lo hace dividiendo todo el espacio de nombres y de IP de Internet en diferentes grupos lógicos. Cada uno de estos grupos tiene autoridad sobre sus propias computadoras y otra información.

Al principio, cuando se formó por primera vez Internet, el número de sistemas en la red era muy pequeño. Era bastante fácil mantener la asignación nombre/dirección. Cada sistema simplemente tenía una lista completa de todos los nombres de sistemas y direcciones en un archivo local. Al acelerarse el crecimiento de Internet el sistema se volvió rápidamente poco manejable. Cuando se agregaba un sistema era necesario actualizar todos los archivos de sistemas. El tamaño de estos archivos comenzó a crecer hasta un gran tamaño. Claramente se necesitaba otra solución.

DNS se dividió conceptualmente en las tres partes siguientes:

- Espacio del nombre de dominio.
- Servidores de nombre.
- Agentes de resolución.

El espacio del nombre de dominio es una especificación de una estructura en árbol que identifica un conjunto de sistemas y suministra información sobre ellos [Rosen, 1997]. Conceptualmente, cada nodo en del árbol tiene una base de información sobre los sistemas bajo su autoridad. Las consultas tratan de extraer la información apropiada de esta base de datos. De forma simple, es el listado de todos los diferentes tipos de información, nombres, direcciones IP, alias de correos, etc. que se pueden consultar en el sistema DNS.

Los programas que guardan y mantienen los datos localizados en el espacio de nombres de dominio se conocen como servidores de nombres. Cada uno de ellos tiene una información completa sobre un subconjunto de espacio de nombres de dominio y tiene información en ante-memoria sobre otras partes. Un servidor de nombres tiene información completa de su área de autoridad. Esta información autorizada se divide en áreas conocidas como zonas que se pueden dividir entre diversos servidores de nombres para suministrar un servicio redundante a una zona. Cada servidor de nombres conoce a otros servidores de nombres que son responsables de diferentes zonas. Si entra una petición de información de una zona de la que es responsable un servidor de nombres determinado, el servidor de nombres simplemente devuelve la información. Sin embargo, si entra una petición en busca de información de una zona distinta, el servidor de nombres contacta con el servidor apropiado con autoridad sobre tal zona.

Los agentes de resolución son simplemente programas o rutinas de biblioteca que extraen información de servidores de nombres en respuesta a una consulta sobre un sistema en el espacio de nombres de dominio.

3.6.3.1 El agente de resolución

El primer paso para usar DNS es configurar la biblioteca del agente de resolución del servidor. Si se quiere usar la resolución de nombres DNS se debe configurar el agente de resolución local, incluso si no se va a ejecutar un servidor de nombres de dominio local.

3.6.3.1.1 El archivo /etc/host.conf

Las bibliotecas de agente de resolución local están configuradas a través de un archivo denominado host.conf que está situado en el directorio /etc. Este archivo informa al agente de resolución sobre qué servicios usar y en qué orden. Este archivo es una simple archivo ASCII que tiene una lista de las opciones del agente de resolución, una por línea. Los campos de este archivo pueden estar separados por espacios o tabuladores. El carácter # indica el comienzo de un comentario. Se pueden indicar varias opciones en el archivo host.conf.

A continuación se muestra un ejemplo de un archivo de configuración /etc/host.conf que usa estas opciones [Trackett, 1998]:

```
# sample /etc/host.conf file
#
# Lookup names vía DNS first then fall back to /etc/hosts

order bind host

# We don´t have machines with multiple addresses

multi off

# check for IP address spoofing

nospoof on

# and warn us if someone attempts to spoof

alert on

# Trim the tristar.com domain name for host lookups

trim tristar.com
```

Este ejemplo muestra una configuración general de agente de resolución para el dominio tristar.com. El agente de resolución consulta los nombres de sistema usando primero DNS y después prueba en el archivo /etc/hosts local.
La posibilidad de múltiples direcciones IP para un solo sistema está desactivada. Este sistema comprueba la falsificación de direcciones IP volviendo a resolver el nombre del sistema que devuelve una consulta de dirección IP inversa. Esto se podría considerar como un sobre esfuerzo, pero ayuda a garantizar que nadie está pretendiendo ser un sistema diferente del que realmente es. Además, se ha configurado el agente de resolución para que avise de un intento de engaño. Finalmente el agente de resolución depura el dominio

trisitar.com de cualquier nombre de sistema que fuese buscado en el archivo /etc/hosts local.

3.6.3.1.2 El archivo /etc/resolu.conf

Ahora ya se ha configurado el comportamiento básico de la biblioteca del agente de resolución, se debe instalar alguna información para la parte DNS del mismo. Sólo es necesario hacerlo si se usa DNS para la resolución de nombres de sistema, es decir, al indicar *bind* en la sentencia *order* del archivo /etc/resolv.conf.

El archivo /etc/resolv.conf controla la forma en la que el agente de resolución usa DNS para resolver nombres de sistema. Indica los servidores de nombres DNS a contactar cuando se resuelve un nombre de sistema y en qué orden se han de contactar. También proporciona el nombre de dominio local y algunas pistas sobre cómo adivinar nombres de dominio de sistemas que se indican sin él.

A continuación un ejemplo del archivo /etc/resolv.conf para tristar.com [Trackett, 1998]:

```
# /etc/resolv.conf for tristar.com
#
# Set our local domain  name

domain tristar.com

# Specify our primary name server

nameserver 166.82.1.3
```

En este ejemplo el usuario indica el dominio local por medio de la opción *domain* y lista un servidor de nombres para resolver los nombres de sistema.
Se ha de indicar la dirección IP del servidor de nombres DNS como un argumento de la opción *nameserver*, no el nombre del sistema. Si se indica el nombre del sistema, DNS no sabe qué sistema contactar para buscar el nombre de sistema de un servidor de nombres.

3.6.4 Servicios de Internet

Si únicamente tuvieramos los mecanismos de intercambio y los servicios de red básicos, un usuario no podría aprovechar la red al máximo, ya que sólo podría acceder a la información de la cual conoce su ubicación y naturaleza, ésta es la razón por la cual surgen los servicios de Internet, los cuales permiten que un usuario de la red acceda a cualquier información sin saber su ubicación fisica.

Existen diferentes sistemas para distribuir la información en Internet, orientados a la explotación de diferentes tipos de información; desde los utilizados para conocer la ubicación de algún archivo del que se conoce el nombre, pasando por sistemas que permiten encontrar información mediante la búsqueda de alguna palabra, hasta los que manejan técnicas de hipertexto y multimedia.

A continuación daremos un breve repaso a los servicios de Internet más útiles y modernos.

3.6.4.1 Archie

El servicio archie es un conjunto de herramientas integradas que proporcionan un directorio electrónico para la localización de información en Internet. Originalmente fue creado para consultar el contenido de servidores de archivos a través de ftp anónimo, pero actualmente el servicio se ha extendido hasta incluir una gran variedad de otros directorios en línea y de listas de otros recursos en Internet. El nombre "archie" se eligió porque archie suena como la palabra "archivo". Por está razón existe la tendencia a hablar de archie como si se tratara de una persona, o al menos de un robot inteligente.

3.6.4.2 WAIS

Los servidores WAIS(Wide Area Information Server) proporcionan otro método de búsqueda de información que se encuentra dispersa por Internet. WAIS permite indizar fácilmente bases de datos distribuidas, aceptando búsquedas a través de palabras clave sencillas o expresiones booleanas complicadas.

3.6.4.3 Gopher

Gopher es el recurso más fácil de usar en Internet, además permite acceder a una gran variedad de información y servicios mucho más que otros recursos en Internet.
Gopher es un potente sistema que permite acceder a muchos de los recursos de Internet de forma simple y consistente. Para usar gopher, todo lo que se necesita es seleccionar en un menú. Cada vez que se hace una selección, gopher hace lo necesario para llevar la petición. Algunos de los elementos de los menús representan otros menús. Si se elige uno de estos elementos, gopher obtiene el nuevo menú y lo despliega, de esta manera se puede pasar de menú en menú usando sólo unas cuantas teclas (o un ratón) para navegar.

3.6.4.4 VERONICA

VERONICA (Very Easy Rodent Oriented Net wide Index of Computerized Archives), es una herramienta que permite mantener la pista de muchos menús de gopher alrededor del mundo. Se puede utilizar "Veronica" para realizar una búsqueda y localizar todas las opciones del menú que contienen ciertas palabras clave (cualesquiera que se especifiquen).

3.6.4.5 World Wide Web

El servicio World Wide Web, también llamado "Web", es una herramienta basada en hipertexto (datos que contienen enlaces o ligas a otros datos, por medio de una interfaz gráfica de usuario GUI, que permite recuperar y mostrar información basada en búsquedas por palabras clave. Lo que hace al servicio World Wide Web tan poderoso es que un enlace o liga puede ir a cualquier tipo de recurso de internet: un archivo de texto, una sesión telnet, un servicio gopher, etc. Este servicio hoy en día es uno de los más populares y más poderosos en cuanto a capacidades, ya que ha pasado a formar parte de un esquema en el

cual se han integrado la mayoría de los servicios incluyendo a los más importantes que son FTP y correo electrónico.

3.7 El SISTEMA OPERATIVO LINUX

Linux es probablemente el acontecimiento más importante del software gratuito desde el original Space War, o más recientemente, Emacs. Se ha convertido en el sistema operativo para los negocios, educación, y provecho personal. Linux ya no es solo para gurús de UNIX que se sientan durante horas frente a la resplandeciente consola (aunque sabemos que un gran número de usuarios pertenece a esta categoría). Linux es un clónico del sistema operativo UNIX que corre en ordenadores Intel desde 80386 en adelante. Soporta un amplio rango de software, desde TEX a X Windows al compilador GNU C/C++ a TCP/IP. Es una implementación de UNIX versátil, distribuida gratuitamente en los términos de la Licencia GNU.

Linux puede convertir cualquier PC en una estación de trabajo. Pone todo el poder de UNIX a nuestro alcance. En los negocios ya se instala Linux en redes enteras, usando el sistema operativo para manejar registros financieros y de hospitales, un entorno de usuario distribuido, telecomunicaciones, etc. Universidades de todo el mundo usan Linux para dar cursos de programación y diseño de sistemas operativos. Y, por supuesto, entusiastas de los ordenadores de todo el mundo están usando Linux en casa para programar, entretenerse, y conocerlo a fondo.

Lo que hace a Linux tan diferente es que es una implementación gratuita de UNIX. Fue y aun es desarrollado por un grupo de voluntarios, principalmente en Internet, intercambiando código, comentando fallos, y arreglando los problemas en un entorno abierto. Cualquiera es bienvenido a sumarse al esfuerzo de desarrollo de Linux: todo lo que se pide es interés en producir un clon gratuito de UNIX y algunos conocimientos de programación.

3.7.1 Descripción general

El desarrollo de Linux es parte de un grupo en expansión de hackers de UNIX que quisieron hacer un sistema con sus propias manos. Existen numerosas versiones de UNIX para muchos sistemas, desde ordenadores personales hasta supercomputadores. La mayoría de las versiones de UNIX para ordenadores personales son muy caras. Una copia para una maquina 386 del UNIX System V de AT&T cuesta unos 1,500 dólares estadounidenses.

Linux es una versión de UNIX de libre distribución, inicialmente desarrollada por Linus Torvalds en la Universidad de Helsinki, en Finlandia. Fue desarrollado con la ayuda de muchos programadores y expertos de UNIX a lo largo y ancho del mundo, gracias a la presencia de Internet. Cualquier habitante del planeta puede acceder a Linux y desarrollar nuevos módulos o cambiarlo a su antojo. El núcleo de Linux no utiliza ni una sola línea del código de AT&T o de cualquier otra fuente de propiedad comercial, y buena parte del software para Linux se desarrolla bajo las reglas del proyecto de GNU de la Free Software Foundation, Cambridge, Massachusetts.

Hoy, Linux es ya un clon de UNIX completo, capaz de ejecutar X Window, TCP/IP, Emacs, UUCP y software de correo y News. Mucho software de libre distribución ha sido ya portado a Linux, y han empezando a aparecer aplicaciones comerciales. El hardware soportado es mucho mayor que en las primeras versiones del núcleo. Mucha gente ha ejecutado pruebas de rendimiento en sus sistemas Linux montados en hardware de PC 486 y se han encontrado que son comparables a las estaciones de trabajo de gama media de Sun Microsystems y Digital. ¿Quién iba a imaginar que este "pequeño" clon de UNIX iba a convertirse en un estándar mundial para los ordenadores personales?

3.7.2 Ventajas y Desventajas

Linux implementa la mayor parte de las características que se encuentran en otras implementaciones de UNIX, más algunas otras que no son habituales. Linux es un sistema operativo completo con multitarea y multiusuario (como cualquier otra versión de UNIX). Esto significa que pueden trabajar varios usuarios simultáneamente en él, y que cada uno de ellos puede tener varios programas en ejecución.

El sistema Linux es compatible con ciertos estándares de UNIX a nivel de código fuente, incluyendo el IEEE POSIX.1, System V y BSD. Fue desarrollado buscando la portabilidad de los códigos fuentes. Siempre encontraremos que casi todo el software gratuito desarrollado para UNIX se compila en Linux sin problemas. Y todo lo que se hace para Linux (código del núcleo, drivers, librerías y programas de usuario) es de libre distribución.

En Linux también se implementa el control de trabajos POSIX (que se usa en los shells csh y bash), las pseudo-terminales (dispositivos pty), y teclados nacionales mediante controladores de teclado cargables dinámicamente. Además, soporta consolas virtuales, lo que permite tener mas de una sesión abierta en la consola de texto y conmutar entre ellas fácilmente. A los usuarios del programa "screen" les resultara familiar esto.

El núcleo es capaz de emular por su cuenta las instrucciones del coprocesador 387, con lo que en cualquier 386 con coprocesador o sin el se podrán ejecutar aplicaciones que lo requieran.

Linux soporta diversos sistemas de ficheros para guardar los datos. Algunos de ellos, como el ext2fs, han sido desarrollados específicamente para Linux. Otros sistemas de ficheros, como el Minix-1 o el de Xenix también están soportados. Y con el de MS-DOS se puede acceder desde Linux a los disquetes y particiones en discos duros formateados con MS-DOS. Además, también soporta el ISO-9660, que es el estándar seguido en el formato de los CD-ROM.

Linux implementa todo lo necesario para trabajar en red con TCP/IP. Desde controladores para las tarjetas de red más populares hasta SLIP/PPP, que permiten acceder a una red TCP/IP por el puerto serie. También se implementan PLIP (para comunicarse por el puerto de la impresora) y NFS (para acceso remoto a ficheros). Y también se han portado los clientes de TCP/IP, como FTP, telnet, NNTP y SMTP.

El núcleo de Linux ha sido desarrollado para utilizar las características del modo protegido de los microprocesadores 80386, 80486 y Pentium. En concreto, hace uso de la gestión de memoria avanzada del modo protegido y otras características avanzadas. Cualquiera que conozca la programación del 386 en el modo protegido sabrá que este modo fue diseñado para su uso en UNIX (o tal vez Multics). Linux hace uso de esta funcionalidad precisamente.

El núcleo soporta ejecutables con paginación por demanda. Esto significa que solo los segmentos del programa que se necesitan se cargan en memoria desde el disco. Las paginas de los ejecutables son compartidas mediante la técnica copy-on-write, contribuyendo todo ello a reducir la cantidad de memoria requerida para las aplicaciones.

Con el fin de incrementar la memoria disponible, Linux implementa la paginación con el disco, puede tener hasta 256 megabytes de espacio de intercambio o "swap" en el disco duro. Cuando el sistema necesita más memoria, expulsa páginas inactivas al disco, permitiendo la ejecución de programas más grandes o aumentando el número de usuarios que puede atender a la vez. Sin embargo, el espacio de intercambio no puede suplir totalmente a la memoria RAM, ya que el primero es mucho más lento que ésta.

La memoria dedicada a los programas y a la cache de disco esta unificada. Por ello, si en cierto momento hay mucha memoria libre, el tamaño de la cache de disco aumentará acelerando así los accesos.

Los ejecutables hacen uso de las librerías de enlace dinámico. Esto significa que los ejecutables comparten el código común de las librerías en único fichero, como sucede en SunOS. Así, los ejecutables serán más cortos a la hora de guardarlos en el disco, incluyendo aquellos que hagan uso de muchas funciones de librería. También pueden enlazarse estáticamente cuando se deseen ejecutables que no requieran la presencia de las librerías dinámicas en el sistema. El enlace dinámico se hace en tiempo de ejecución, con lo que el programador puede cambiar las librerías sin necesidad de recompilación de los ejecutables.

3.7.3 Distribuciones Linux

Al ser Linux un software de libre distribución, no hay ninguna organización o entidad responsable de mantenerlo y distribuirlo. Por lo tanto, cualquiera es libre de agrupar y distribuir el software, en tanto respete las restricciones de la GPL. El resultado final de esto es que existen muchas distribuciones de Linux, disponibles a través de FTP anónimo o pidiéndolo por correo.

Alguna vez nos encontraremos con la tarea de decidir por una distribución en particular de Linux que se ajuste a las necesidades. No todas las distribuciones son iguales. Muchas de ellas incluyen prácticamente todo el software que se necesitaría para poner en marcha un sistema completo y algunas otras distribuciones son "pequeñas" distribuciones orientadas a usuarios sin copiosas cantidades de espacio en disco. Muchas distribuciones solamente contienen lo esencial del software de Linux, y se espera que el usuario instale por su propia cuenta paquetes de software más grandes, como el Sistema X-Window.

El Linux Distribution HOWTO que se puede obtener en Internet contiene una lista de las distribuciones de Linux disponibles a través de Internet, así como por correo.
¿Cómo podemos decidir entre todas estas distribuciones? Si se tiene acceso a las news de USENET, u otro sistema de conferencias por computadora, podríamos preguntar allí las opiniones personales de la gente que haya instalado Linux. Incluso mejor, si conoce a alguien que haya instalado el Linux, podemos pedir consejo y ayuda. Hay muchos factores a considerar cuando se elige una distribución, sin embargo, las necesidades y opiniones de cada uno son diferentes. En la actualidad, la mayoría de las distribuciones populares de Linux contienen aproximadamente el mismo conjunto de software, de forma que la elección de una distribución es más o menos arbitraria. Algunas de las más comunes son: Red Hat, Slackware, S.U.S.E. Caldera Open Linux, Debian y otras más.

3.7.4 Instalación de Linux

Para correr Linux se necesita al menos 4 Mb de memoria en la máquina, aunque se recomienda 8 Mb si se tiene pensado utilizar X-window, y 16 Mb si lo que se quiere es utilizar Linux en entorno de redes; pero como consejo válido para cualquier sistema operativo, entre más memoria posea el sistema, más feliz se puede operar en él. Obviamente se necesita un disco duro y un controlador estándar tipo AT. Todas las unidades MFM, RLL e IDE y controladores deben funcionar. Muchas unidades y adaptadores SCSI también están soportados. También se necesita una unidad de disco 3 ½ o 5 ¼ pulgadas, la cual utilizaremos para la instalación y el mantenimiento del sistema.

La cantidad de espacio libre en el disco duro necesaria depende de la cantidad de software que se pretenda instalar. La mayoría de las instalaciones requieren entre 40 y 100 Mb como mínimo. Esto incluye espacio para el software para realizar swap (uso del disco duro como memoria virtual) y espacio libre para los usuarios. Sin embargo nosotros elegimos el límite, pueden ser 10 Mb o 500, tal vez más, depende de lo que queramos hacer e instalar.

Linux puede coexistir con otros sistemas operativos, como MS-DOS, Windows 9.x u OS/2. En el disco duro, de hecho se instalan emuladores para estos sistemas operativos e incluso respeta la organización de archivos de cada uno de ellos, tanto la fat-16 de MS-DOS, fat-32, la HPFS de OS/2 y muchos estructuras más.

También se necesita una tarjeta de vídeo Hércules, CGA, VGA o SVGA y un monitor. Por término general si se es capaz de ver algo con la tarjeta de vídeo y el monitor bajo MS-DOS, también se puede bajo Linux sin lugar a duda. Sin embargo puede que se tengan problemas para el ambiente gráfico X-Window, pero si se cuenta con elementos estándar como monitores entrelazados o no entrelazados, tarjetas Cirrus, Trident, etc. podemos confiar en que no habrá problemas al respecto.

Prácticamente el 95 % de todas las computadoras de hoy en día pueden ejecutar Linux sin problemas de ningún tipo, todo depende del trabajo que se quiera desempeñar. Incluso con un equipo no tan alejado de nuestras posibilidades, en concreto un 386 puede ejecutar Linux correctamente.

3.7.4.1 Instalación de RedHat Linux

RedHat es una de las distribuciones de Linux más importantes en el mundo entero, en realidad la empresa RedHat al igual que muchas otras empresas que se encargan de distribuir Linux, han desarrollado ya programas que se encargan de facilitar la instalación de Linux en las computadoras, además de agregar muchas utilerías y programas adicionales que permiten al usuario o administrador tomar ventaja de dichos programas para manejar con mayor facilidad el sistema Linux.

Describiremos aquí el proceso de instalación de RedHat Linux por ser una de las distribuciones más comunes y con mayores facilidades para el usuario.

3.7.4.1.1 Como comenzar

El primer paso es conocer el hardware de donde se va a instalar Linux. Verificar que todos los componentes sean compatibles con Linux, ya que esto evitará dolores de cabeza innecesarios.

Los requisitos mínimos para la instalación de un sistema Linux se describieron anteriormente, pero los mencionaremos de manera breve y se listan a continuación.

- Procesador 386SX o mayor.
- 8 Mb de RAM.
- Una tarjeta de vídeo soportada.
- Una unidad de CDROM soportada.
- Unidad de disco flexible de 3 ½ o 5 ¼.
- Espacio en disco suficiente para el tipo de instalación (una instalación típica consume alrededor de 250 Mb).
- Mouse (si se desea utilizar el modo gráfico).

Seguramente debemos contar ya con una edición de Linux RedHat ya sea que la hayamos obtenido por algún medio como ftp, CD-ROM, etc. Generalmente todas las distribuciones de Linux cuentan con un documento llamado Hardware-HOWTO, el cual es una referencia del hardware soportado por Linux. Este archivo es de tipo texto y viene comprimido. Para descomprimirlo se utiliza la utilería gzip.exe que viene incluida también en la distribución.

Si el sistema al que pretendemos instalar Linux esta conectado a red, se necesitará además conocer la dirección IP del equipo, la dirección de red, la máscara de subred, la dirección del ruteador, la dirección del servidor de nombres y el nombre que tendrá la máquina Linux en el DNS.

Linux es capaz de coexistir con otros sistemas operativos. Para esto se debe particionar el disco duro según las necesidades de los otros sistemas operativos y Linux. Si ya se tiene algún sistema MS-DOS en el equipo y no se desea borrarlo para particionar el disco, se puede utilizar la utilería fips, la cual también se distribuye junto con Linux, sin embargo si se cuenta con un CD-ROM, se pueden buscar las utilerías en el directorio /dosutils del CD junto con la documentación. Para usar fips se debe leer con mucho cuidado la

documentación antes utilizarlo, pues no se ofrece ninguna garantía en caso de perder los datos del disco.

3.7.4.1.1 Discos de Arranque

Dependiendo de la manera en que se instale Linux RedHat, se necesitará crear uno o dos discos flexibles de 3 ½ pulgadas. Los archivos imagen se encuentran en el subdirectorio /images del CD o en un directorio similar si no se tiene CD. Si se instala por CD-ROM o por NFS solamente se necesita el boot.img, pero si se usa cualquier otro método de instalación, o se tienen dispositivos PCMCIA, se necesitará el disco suplemento supp.img.

Los disquetes deben estar formateados y libres de errores. Bajo MS-DOS se debe usar el programa rawrite.exe, que se encuentra en el subdirectorio /dosutils. Ejecutar rawrite y seguir las instrucciones de uso. Si se crean desde un sistema Linux se usa el comando dd:

```
dd if=boot.img of=/dev/fd0 bs=1440 k
```

3.7.4.1.2 Instalación con CD-ROM

Es importante que el CD-ROM sea compatible con Linux. Si no reconoce el CD-ROM, el archivo CDROM-HOWTO y los manuales podrían aportar alguna pista. A veces los CD-ROM que vienen conectados al puerto IDE de la tarjeta de sonido no son reconocidos. Si este es el caso, es recomendable probar conectando la unidad de CD como esclavo, al mismo puerto IDE del disco duro. También se puede intentar el método de instalación desde disco duro, copiando previamente el subdirectorio /redhat del CD a una partición ya existente en el disco duro.

Ahora hay que insertar el disquete de arranque que se creó con boot.img, el CD en su respectiva unidad, encender la máquina y seguir las instrucciones.

3.7.4.1.3 Durante la instalación

Durante la instalación se tiene a disposición cinco terminales virtuales que proporcionan información sobre la instalación. Se pueden invocar presionando ALT + Fn, siendo Fn una de las teclas de función, de la F1 a la F5.

F1 Pantalla de Instalación.
F2 Shell.
F3 Bitácora de Instalación.
F4 Bitácora del Sistema.
F5 Salida de los programas en ejecución.

A la pregunta de que si se quiere soporte para PCMCIA se responde NO al menos que se tenga algún dispositivo PCMCIA.

A continuación se pregunta desde donde se instalará Linux. Las opciones son CD-ROM, NFS, FTP o disco duro. Elegimos entonces la opción de CD-ROM en este caso y a

continuación se pedirá que se indique el tipo de CD, IDE/ATAPI, SCSI u otro. Si el CD-ROM no es SCSI o no se está seguro, se puede comenzar con la opción IDE. Si se elige OTHER, debemos pedir al sistema que ejecute una autoprueba. Si se recorren todas las opciones y no es reconocido el CD-ROM, hay que checar todas las conexiones, probablemente de fábrica el CD-ROM venga conectado a la tarjeta de sonido, por lo que se deberá conectar como esclavo a la tarjeta IDE de donde está el disco duro.

Después el sistema preguntará si se quiere actualizar el sistema o instalar un sistema nuevo. Como se está instalando Linux por primera vez, se debe elegir INSTALL.

3.7.4.1.4 Particiones de disco

Para instalar Linux se necesita crear como mínimo dos particiones. Una de ellas será dedicada a la partición SWAP (de memoria virtual) del sistema Linux. Se recomienda que si se tiene menos de 16 Mb de memoria RAM, se utilice un tamaño de por lo menos 16 Mb. Si se tiene más, se puede asignar como memoria virtual el mismo tamaño de la RAM. En esta opción, el sistema mostrará las unidades de disco disponibles en la máquina. hay que recordar que Linux y UNIX consideran los dispositivos como directorios, por lo que probablemente se vea algo como esto:

```
/dev/hda
/dev/hdb
/dev/sda
```

Dónde /dev/hda se refiere a la primera unidad, maestra, del sistema (en MS-DOS sería unidad C), /dev/hdb se refiere a una unidad esclavo (unidad D en MS-DOS). /dev/sda tiene el mismo significado, solo que se utiliza para unidades SCSI. Entonces se escoge la unidad en donde se instalará Linux y presionamos EDIT. Se iniciará el programa fdisk (en las nuevas versiones de RedHat es posible utilizar programas opcionales a fdisk), que permite hacer las particiones, los comandos son:

m imprime un menú con las opciones básicas.
p muestra en pantalla el estado de las particiones.
n crea una nueva partición (deberá ser primaria).
t cambia el tipo de partición (por ejemplo 82 para una partición swap de Linux y 83 para Linux nativo).
w guarda los cambios y sale de fdisk.
q sale de fdisk sin guardar los cambios.

Una configuración típica podría ser:

```
Disk /dev/hda: 64 heads, 63 sectors, 524 cylinders
Units = cylinders of 4032 * 512 bytes

Device Boot   Begin  Start  End   Blocks  Id      System
/dev/hda1     *      1      1     102     205600  Dos
/dev/hda2     103    103    117   30240    82     swap
/dev/hda3     *      118    118   524     820512  native
```

Si se tiene más de un disco duros se quiere tener las cosas por separado, se pueden crear particiones para usuarios, para programas, etc.

Escogemos entonces cual es la partición de root, que en el ejemplo es la única partición nativa de Linux y escogemos si se quiere montar otras particiones. En el ejemplo la partición /devb/hda1 se montó en /mnt/dos, para poder usarla desde Linux.

El siguiente paso consiste en escoger los componentes del sistema Linux. Una instalación completa abarca de 350 a 500 Mb, una instalación básica unos 150 Mb y una Instalación típica (X-Window incluido) unos 200 Mb. Se escoge de la casilla y se seleccionan los paquetes individuales que se muestran para instalar. Esto desplegará en ventanas el contenido de cada conjunto de componentes. Tecleando F1 se obtiene una descripción de cada paquete. Si no se está seguro de que componentes se deben instalar, no hay que preocuparse, más adelante se puede usar el programa rpm o el glint para administrar los componentes.

3.7.4.1.5 Configuración del ratón

Ahora se debe escoger el tipo de ratón con que cuenta el sistema. Si se tiene un ratón con entrada minidin, se escoge el mouse PS/2. Si se tiene un con dos botones, escogemos la opción para emular 3 botones. Si se tiene un ratón serial, indicamos el puerto serial (COM) donde está conectado.

3.7.4.1.6 Configuración del sistema X-Window

Para configurar el sistema X Window, se selecciona el tipo de vídeo que se tiene. Si no se está seguro es posible elegir la opción genérica (unlisted card). El siguiente paso es seleccionar el monitor. Si el monitor no está listado seleccionamos custom e ingresamos los parámetros correspondientes. Si no se está seguro seleccionamos el modelo VGA genérico. Elegimos la cantidad de memoria de tarjeta de vídeo. El siguiente paso pregunta sobre el Clockchip. Es recomendable que se escoja el default. Para finalizar se escogen los modos de vídeo que se quieran utilizar.

3.7.4.1.7 Configuración de red.

Si el equipo está conectado a una red, habrá que averiguar la dirección IP, el nombre DNS del equipo, el gateway, y la máscara de subred así como la dirección IP del servidor de nombres. Al tener esta información llenamos los campos que se solicitan. Si no se está conectado a la red pasamos por alto este paso.

3.7.4.1.8 Configuración del reloj

Podemos escoger entre GMT y UTC. UTC es el estándar internacional. Si se está conectado a una red, escogemos UTC. Si no es el caso, seleccionamos GMT y el país/localidad en donde estamos (México en esta caso), hay tres zonas para México, escogemos la

correspondiente a la localidad. La opción América (mexico_city incluye el ajuste para el horario de verano, por lo que es la más recomendable para la mayor parte del país.

3.7.4.1.9 Selección del teclado

Si se va a utilizar el teclado en inglés, seleccionamos US style keyboard. Si en cambio tenemos uno con distribución de teclas en español, seleccionamos la opción ES style keyboard.

3.7.4.1.10 Password de root

Asignamos un password al administrador del sistema (root). Escogemos uno que no se olvide pero que tampoco sea obvio.

3.7.4.1.11 Instalación de LILO

Linux Loader es el programa que permite inicializar la máquina y escoger entre 2 o más sistemas operativos que se tengan instalados en el equipo. Hay que elegir para su instalación MBR (master boot record), a menos que se tenga algún requerimiento específico. Es muy seguro y recomendable instalar LILO.

Finalmente sacamos el disco flexible de la unidad de disquete, reiniciamos el equipo y a trabajar.

Capítulo 4

Los Sistemas Operativos como Plataformas Estratégicas

4.1 SISTEMAS OPERATIVOS, FUNCIONES BÁSICAS

Los sistemas operativos de redes son uno de los componentes más importantes y de más rápida evolución en la industria de las Tecnologías de Información (TI). Cada computadora tiene un sistema operativo, el cual consiste en un conjunto de programas que desempeñan funciones básicas para el manejo de procesos, memoria, dispositivos y archivos.

Las primeras versiones de sistemas operativos de redes para servidores establecieron las siguientes expectativas sobre los servicios mínimos que los sistemas operativos para redes debían ofrecer:

- Permitieron a los clientes guardar y tener acceso a sus archivos de datos en el disco duro de un servidor, y enviar archivos hacia las impresoras del servidor.

- Protegieron los archivos de un usuario contra acceso sin autorización, ya que el administrador podía definir los directorios del servidor que cada usuario o grupo de usuarios tenía permiso de utilizar.

- Dieron a los administradores herramientas básicas para analizar la carga que los usuarios ponían sobre los recursos del sistema, y localizaron obstrucciones que afectaban el buen funcionamiento del sistema en general.

- Ofrecían sistemas de mensajes rudimentarios que permitieron a los clientes comunicarse entre sí.

Sin embargo, conforme ha crecido el papel de las redes en las organizaciones, se ha requerido una nueva serie de características para servidores:

- Respaldo para una diversidad de redes más grandes, el funcionamiento en una variedad de servidores, el manejo de diferentes protocolos de transporte, el reconocimiento de múltiples tipos de NICs (Network Interface Controllers / Controladores de interfase de red), y el respaldo para un amplio rango de hardware y sistemas operativos de PCs cliente.

- Servicios poderosos de nombres y directorios que ofrezcan a los usuarios un método estándar y fácil de usar para tener acceso a los recursos en estas redes altamente diversas.

- Manejo de aplicaciones de bases de datos para PCs cliente y servidor. Conforme los recursos de estas aplicaciones de bases de datos requieren ser aumentados, los sistemas

operativos de red pueden necesitar dar apoyo a múltiples CPUs en el mismo servidor, o manejar procesamiento distribuido por servidores múltiples.

- Proveer o manejar aplicaciones que ofrecen poderosos servicios de mensajes internos.
- Manejo de servicios de acceso telefónico para usuarios remotos.
- Tolerancia a fallas, incluyendo el reflejo y/o duplicación de drives y servidores completos, y control de fuentes de poder de respaldo inteligentes, para mantener en disponibilidad los recursos del servidor que son cada vez más vitales al enfrentar una falla de componentes esenciales.

Actualmente, los sistemas operativos cuentan con capacidades básicas integradas para redes. Esto facilita la conexión de PCs a servidores LAN (Local Area Network; Red de área local) para que compartan archivos y la impresora, así como la construcción de redes "punto a punto" simplemente conectando las PCs de escritorio a los cables. También es posible usar sistemas operativos integrados para conectarse con sitios remotos vía modems, sin tener que utilizar otras aplicaciones.

4.2 ¿HACIA DÓNDE VAN LOS SISTEMAS OPERATIVOS?

Mientras que Internet y las tecnologías que lo acompañan se han convertido cada vez más en partes importantes de una estrategia de redes global, varios componentes de software de Internet se han ido integrando a los sistemas operativos de redes estándar, incluyendo servidores HTTP (HyperText Transfer Protocol; Protocolo de transferencia de textos), servidores FTP (File Transfer Protocol; Protocolo de transferencia de archivos) y Gopher.

Las versiones originales de todos estos servicios fueron desarrolladas para UNIX, el sistema operativo original de Internet, pero debido a la importancia actual del Internet, UNIX ha comenzado a ser no sólo una opción para instituciones científicas y académicas, sino que se ha convertido en un sistema operativo para el manejo de servidores de alto volumen, servicios de WWW y aplicaciones de bases de datos.

UNIX presenta una tendencia hacia las funcionalidades del gran centro de datos, ambientes más controlados, mejor administrados, y más seguros. Se enfoca hacia la búsqueda de funcionalidades que permitan tolerar fallas de componentes y reforzar la disponibilidad del sistema en cualquier momento.

De los sistemas operativos para redes comerciales, Novell Netware tiene una gran participación en el mercado de servidores de archivos e impresión, mientras que Windows NT se ha convertido en el sistema operativo líder para servidores de aplicaciones de oficina. No obstante, todos estos productos están siendo objeto de modificaciones para incluir más servicios de Internet, y podemos esperar que todos lleguen a manejar la conectividad y los servicios de Internet de manera integrada. La navegación en Internet es una de las razones más importantes para comprar una PC; sin embargo, un número cada vez más grande de dispositivos se estará conectando a la red. Una Televisión, una videocasetera, un teléfono, lavadora, un refrigerador, un horno de microondas, un estéreo,

etc. Actualmente todos los dispositivos contienen un microprocesador, pero no todos pueden conectarse a Internet.

Los sistemas operativos tenderán a realizar funciones más especializadas y a convertirse en plataformas conectadas a la red. Asimismo, serán sistemas operativos más simples y sencillos, y estarán orientados a las necesidades del usuario, sin dejar de lado el cumplimiento con los estándares de comunicación y de programación.

La oblicuidad es una tendencia también importante que se vislumbra. Esto significa que cualquier persona será capaz de manejar una computadora, ya que éstas se volverán tan simples como los teléfonos y solamente nos brindarán un servicio. La tendencia será acceder servicios, por lo que requeriremos sistemas operativos sencillos que ya estén en red y sólo esperen el Web Tone (tono de red) para accederlos, siendo totalmente transparentes para el usuario.

Las tendencias de los sistemas operativos de red se dirigen hacia la escalabilidad (capacidad del sistema operativo de aumentar su capacidad de servicio sin disminuir su calidad), la alta disponibilidad (sistemas fiables que puedan correr las 24 horas de los 365 días del año), el soporte de aplicaciones distribuidas, la integración de servicios de comunicación, y la reducción del costo total de propiedad TCO (por sus siglas en ingles *Total Cost Ownership*) consecuencia de una fácil administración, sencillez de uso, fácil configuración, y autodetección de hardware.

4.4 ¿CUÁL SERÁ LA TENDENCIA?

Los sistemas operativos tenderán a realizar funciones más especializadas y a convertirse en plataformas conectadas a la red Internet. Asimismo, serán sistemas operativos más simples y sencillos, y estarán orientados a las necesidades del usuario, sin dejar de lado el cumplimiento con los estándares de comunicación y de programación.

La oblicuidad es una tendencia también importante que se vislumbra. Esto significa que cualquier persona será capaz de manejar una computadora, ya que éstas se volverán tan simples como los teléfonos y solamente nos brindarán un servicio. La tendencia será acceder servicios, por lo que requeriremos sistemas operativos sencillos que ya estén en red y sólo esperen el Web Tone (tono de red) para accederlos, siendo totalmente transparentes para el usuario.

Las tendencias de los sistemas operativos de red se dirigen hacia la escalabilidad (capacidad del sistema operativo de aumentar su capacidad de servicio sin disminuir su calidad), la alta disponibilidad (sistemas fiables que puedan correr las 24 horas de los 365 días del año), el soporte de aplicaciones distribuidas y la integración de servicios de comunicación.

La tendencia de los sistemas operativos está orientada a la centralización de los servidores y a los clientes con el fin de controlar la red desde el servidor, mientras que los usuarios accesan sólo a las aplicaciones que les fueron asignadas. La comunicación debe ser abierta

y basada en estándares, como Java. Esto significa que los clientes podrán desarrollar nuevas aplicaciones sobre Java que puedan correr en cualquier plataforma operativa.

Hay una tendencia de los sistemas operativos a soportar las fallas dentro de máquinas independientes. El desempeño de los sistemas operativos también tenderá a mejorar, así como la integración de los sistemas operativos con los sistemas de administración. Estos van a empaquetarse para ofrecer soluciones, y habrá máquinas cada vez más especializadas.

En cuanto a la funcionalidad se visualiza una clara tendencia a optimizar la que actualmente existe, que los sistemas operativos sean más rápidos, más sencillos, más autoadministrables, para que puedan integrarse a otros sistemas operativos diferentes. La tendencia de los mercados no es comprar equipo ni sistemas operativos, sino comprar servicios y funcionalidad, clave del éxito de una empresa.

4.5 VENTAJAS COMPETITIVAS EN SISTEMAS OPERATIVOS MODERNOS

4.5.1 Nuevos servicios en Windows 2000

Microsoft Windows NT Server está optimizado para soportar hasta 4 Gb de memoria RAM y cuenta con multiprocesamiento simétrico, lo que permite agregar hasta 32 procesadores y manejar discos duros de 16 Tb. El sistema operativo de Microsoft integra la tecnología Wolf Pack o clusters para agrupar múltiples servidores NT, los cuales comparten recursos (disco duro, memoria y aplicaciones). Si alguno de estos equipos falla, otro de los servidores en funcionamiento restablece los enlaces automáticamente para que la tarea que estaba ejecutándose no se pierda.

Para realizar el trabajo colaborativo, los sistemas operativos de redes tienen el reto de habilitar el desarrollo de aplicaciones basadas en Internet. Para lograr la integración de las aplicaciones, Windows NT Server integra el modelo DIA (Distributed Internet Applications; Aplicaciones de Internet distribuidas), el Microsoft Message Queue Server y el Microsoft Transaction Server. A través de estos componentes, los desarrolladores pueden hacer drag and drop desde el código de su aplicación hacia Transaction Server, el cual convierte la aplicación cliente/servidor en una aplicación distribuida, utilizando componentes distribuidos en múltiples sitios. Esto reduce hasta en un 60 por ciento el tiempo de desarrollo de las aplicaciones distribuidas.

Los sistemas operativos de red también tienen el reto de satisfacer las múltiples y diversas necesidades de los usuarios, desde compartir un archivo hasta realizar una compra por Internet. Por eso Windows 2000 integra Quality of Service (QoS; Calidad de servicio), la cual brinda inteligencia al sistema operativo para distinguir paquetes de información y asignarles automáticamente el ancho de banda adecuado para que lleguen a su destino de manera óptima.

Windows 2000 Server también integra diferentes tecnologías para realizar comercio electrónico seguro: Kerberos, algoritmos de encriptación, SSL (Secure Socket Layer; Capa de receptáculo seguro), SET (Secure Electronic Transaction; Transacción electrónica segura), e interfases de comercio electrónico. IntelliMirror también es un nuevo servicio de

Windows 2000, el cual permite asociar una imagen de las aplicaciones que necesita cada usuario de la red, a través de su perfil. De esta manera cada usuario tendrá sus aplicaciones a la mano, sin importar que no se encuentren instaladas en la máquina que esté utilizando.

Los servicios de directorio activo (ADS; Active Directory Services), son una nueva adición a Windows 2000. Los servicios de directorio permiten integrar más información de los objetos de la red (usuarios, impresoras, faxes y componentes lógicos de una aplicación) y agregarles propiedades, extendiendo de esta manera el directorio. Por lo tanto, el sistema de directorio permite administrar millones de objetos desde un solo lugar.

4.5.2 Novell

Los servicios de directorio de Novell (NDS), continúan siendo una tecnología fundamental en la última versión de Netware; sin embargo, otras características como su orientación hacia los servicios basados en Internet y la adición de IP (Internet Protocol; Protocolo de Internet) nativo también son importantes.

Novell Storage Services (NSS; Servicios de Almacenamiento de Novell), otro elemento crucial de Netware 5, permite guardar grandes volúmenes de información aproximadamente hasta 8 Tb en menos de 10 segundos. Esto significa que Netware 5 es un ambiente ideal para empresas con aplicaciones de misión crítica. Además, es escalable y abierto, puesto que permite la convivencia de diferentes ambientes como UNIX o Windows NT, los cuales son administrados a partir de NDS y está por liberarse NDS para Linux, y se encuentra en desarrollo NDS para AS/400 y mainframe, en conjunto con IBM.

La administración del escritorio también es un proceso importante de una red, por lo que uno de los componentes de Netware 5, llamado Z.E.N. Works, permite distribuir software de manera automática y configurar el ambiente de trabajo de acuerdo al perfil de cada uno de los usuarios, o de acuerdo al perfil del equipo.

Netware 5 está dirigido a aquellas empresas o instituciones que tienen la necesidad de servicios globales e Internet, desde los servicios básicos como son compartir archivos e impresoras, hasta la integración de sistemas de colaboración o de publicación en Internet, aunque manejen plataformas heterogéneas.

4.5.3 Solaris

Solaris 8, la última versión del sistema operativo de Sun Microsystems, es un UNIX de 64 bits que cumple con los estándares de la industria, por lo que está certificado como UNIX 98. Solaris puede correr sobre sistemas SPARC e Intel, y permite manejar tráfico de red pesado y grandes cantidades de información.

Las aplicaciones que corren bajo Solaris pueden direccionar los datos directamente desde la memoria en lugar de hacerlo desde el disco, así que las operaciones que generalmente utilizan mucho tiempo, pueden ejecutarse en fracciones pequeñas. Además, una mayor cantidad de procesos pueden correr simultáneamente, lo que permite a los servidores individuales manejar más aplicaciones.

Algunas características de Solaris como reconfiguración dinámica, y balanceo de cargas, permiten minimizar la recuperación de la velocidad del sistema. Además, Solaris cumple con todos los estándares para interoperar con cualquier número de máquinas y de usuarios a escala mundial.

En cuanto a la seguridad del sistema, Solaris ofrece seguridad C2 (pudiendo alcanzar seguridad B1). Ejemplos del nivel de seguridad que ofrece Solaris, fue la alta seguridad que ofreció Solaris en las elecciones para gobernador del D.F. en 1997, y la implementación de este sistema por el Banco de México.

4.5.4 OS/2 de IBM

Actualmente muchos usuarios tienen instalado un sistema operativo en su PC, y pueden instalar nuevas aplicaciones o programas mediante CDs o diskettes, modificando la configuración de su máquina. Esto hace que el administrador pierda el control total de la red.

Bajo este esquema, OS/2 surge como un sistema para aplicaciones de misión crítica, apoyando la estrategia de Network Computing de IBM, puesto que está basado en cliente/servidor. De esta manera, una empresa puede tener un servidor robusto con OS/2 Warp Server for E-Business y varias PCs sin disco duro, las cuales se conectan a la red para bajar la información y las aplicaciones desde el servidor.

Las herramientas como WorkSpace On-Demand, instalada sobre Warp Server for E-Business, permite que el servidor envíe a la PC del usuario una imagen de las aplicaciones de acuerdo a su perfil. Así, el usuario sólo accesa las aplicaciones que le fueron asignadas para realizar su trabajo, mientras que el administrador puede tener un mayor control de los recursos y las aplicaciones desde el servidor, reduciendo los costos de mantenimiento de la red.

OS/2 es una plataforma escalable y puede convivir con diferentes ambientes como UNIX, mainframes o AS/400, soportando TCP/IP para realizar la comunicación. Además, soporta los estándares de la industria encaminados hacia Java y el Network Computing, a diferencia de sistemas como Windows NT, que todavía sigue siendo un esquema tipo cliente/servidor.

OS/2 Warp Server permite compartir impresoras y archivos, y proporciona sistemas de administración de respaldos, recuperación de conexiones y acceso a Internet. Es escalable hasta 64 procesadores, proporcionando seguridad y disponibilidad a muchas instituciones bancarias, las cuales utilizan este sistema operativo para mantener la operación de su negocio.

4.5.5 SCO UNIX

UNIXWARE 7, el sistema operativo de SCO, es un sistema operativo UNIX abierto que corre sobre la plataforma Intel o similares, es decir, otros procesadores compatibles con

Intel, como AMD y Cyrix. Además, UNIXWARE 7 será uno de los primeros sistemas operativos que trabajará a 64 bits.

El sistema operativo de SCO cumple con las especificaciones de UNIX 95, y es escalable (hasta 32 procesadores, 64 Gb de memoria RAM y 76800 Tb de disco duro externo). Combinado con la tecnología de clustering de SCO Reliant HA, UnixWare 7 ofrece clusters de alta disponibilidad y tolerancia a fallas.

Este sistema UNIX es fácil de usar, ya que utiliza una interfase gráfica y amigable llamada Webtop, por lo que su administración es sencilla desde cualquier lugar, a través de cualquier dispositivo que soporte un browser (navegador).

UNIXWARE 7 soporta también las bases de datos y aplicaciones empresariales más populares como Oracle, Informix y Baan, y los protocolos de Internet. Está diseñado para las aplicaciones críticas de las grandes empresas y puede integrarse con facilidad en sistemas heredados.

En cuanto a seguridad, UNIXWARE de SCO brinda seguridad de nivel C2, con perfiles predefinidos a elegir, desde el nivel máximo hasta el mínimo. Integra, además, registros de auditoría, contabilidad de accesos al sistema, identificación, autenticación, y control de accesos.

UNIXWARE puede funcionar como servidor en una máquina con 32 Mb en RAM, un disco duro de 1 Gb y un procesador 486 o Pentium, lo que significa que es un sistema operativo ligero y poco demandante de recursos en hardware.

4.5.6 El Unix de libre distribución

Linux es un sistema operativo libre, es decir, puede compartirse y utilizarse sin restricciones, lo que permite a los desarrolladores crear y mejorar las aplicaciones rápidamente. Este sistema está disponible para plataformas Intel, Silicon Graphics, SPARC, Alpha, RS6000, Power PC para Macintosh, y algunas distribuciones en ciernes para HP9000.

Linux es un sistema operativo para red, porque su enfoque principal es como servidor. Prueba de ello es que muchos fabricantes están vendiendo servidores con Linux preinstalado. Dell, Compaq, IBM, Silicon Graphics y Sun son algunas empresas que venden computadoras preinstaladas con Linux.

La versión actual del Kernel de Linux es 2.2.10, y entre sus aplicaciones se encuentran servidores de Web, servidores de archivos e impresoras, servidor de correo, y aplicaciones de escritorio así como la suite StarOffice de Sun Microsystems, que incluye un procesador de palabras, una hoja electrónica, presentaciones y correo electrónico. Linux también puede ser utilizado para realizar gráficos en tercera dimensión.

Debido a que Linux es un software libre, los problemas de seguridad son detectables y reparados rápidamente, por lo que resulta seguro y es muy utilizado en firewalls (paredes

de fuego). Asimismo, Linux puede utilizarse como servidor de base de datos. Un ejemplo de esta aplicación es el sistema de información de accidentes y el programa de auditorías internas de la Comisión Federal de Electricidad (CFE) en México, los cuales se encuentran en una base de datos Informix con Linux.

Como ejemplos de la utilización del sistema operativo Linux destacan los trenes del Norte de Italia, los cuales son controlados por computadoras con Linux; el sistema del Hipódromo en Australia también está basado en 500 puntos de conexión de Linux debido a su estabilidad y fácil administración; los servidores interno y externo, y una parte del firewall de la CFE son Linux también [Welsh, 1999]. El Archivo General de la Nación es otra organización que está utilizando Linux. Por su parte, la empresa de telefonía llamada Grupo Sitel utiliza un firewall con Linux, así como Acer Latinoamérica.

Aunque Linux trabaja por medio de comandos, últimamente se han desarrollado muchos ambientes gráficos para éste sistema como KDE y Gnome. Por ejemplo, Netscape tiene versiones para Linux, los cuales corren sobre una interfase gráfica. Asimismo, Linux tiene integrada la característica de "Calidad del Servicio", es decir, la administración del ancho de banda de acuerdo a los paquetes de información. También incluye servicio de directorio, servicio de nombres (DNS) y clustering.

4.6 ELEGIR EL SISTEMA OPERATIVO ADECUADO

Para elegir un sistema operativo, es importante tener en cuenta el tamaño o el tipo de cliente que lo va a utilizar y para qué va a utilizarse. Los ambientes de operación de un sistema operativo pueden variar: dispositivos conectados a una televisión, pequeños sistemas operativos impresos en un chip para operar dentro de un ruteador de red, sistemas operativos para PCs, sistemas operativos para equipos portables, sistemas para estaciones de trabajo de uso rudo o de alto volumen de tráfico de datos, sistemas para el manejo de redes, o sistemas operativos para aplicaciones de misión crítica.

Respecto al tipo de uso, existen sistemas operativos para uso doméstico (sencillos de usar y de fácil configuración, con múltiples asistentes para el apoyo de los usuarios) hasta sistemas operativos de ambientes corporativos, necesarios para correr aplicaciones de misión crítica, en donde la seguridad, la escalabilidad, el bajo costo total de propiedad y la alta disponibilidad, son factores cruciales.

Como se puede apreciar, existen múltiples factores bajo los cuales los sistemas operativos son diseñados. Si se intenta correr en una estación de trabajo un sistema diseñado para correr en un Web TV, su desempeño no sería el adecuado. Por eso, es importante que se tome en cuenta que hay una gran variedad de sistemas operativos para satisfacer todas las necesidades de información del usuario.

Por esta razón, no debemos valorar lo que hace un sistema operativo, sino identificar cuáles son sus las necesidades de los usuarios para elegir el sistema que las cubra adecuadamente, es decir, se debe adquirir la tecnología que realmente le traerá beneficios y hará crecer el negocio, y dividir claramente los beneficios adicionales de sus requerimientos primordiales.

Recordemos que no existe un sistema operativo que por sí mismo ofrezca una solución total. Nadie compra un sistema operativo sin hacer un análisis previo. El sistema operativo realiza ciertas funciones y las aplicaciones cumplen otras; juntos generan una solución. El sistema operativo de red debe ayudar en la ejecución de las aplicaciones, obteniendo el mejor provecho posible del hardware.

Funciones que realiza un sistema operativo

- Definir la interfase sistema-usuario.
- Interpreta comandos.
- Administra los procesos del sistema.
- Controlar y compartir los recursos de hardware entre los usuarios.
- Permitir a los usuarios compartir sus datos entre ellos.
- Prevenir que las actividades de un usuario no interfieran en las de los demás usuarios.
- Organizar los datos de tal manera que se pueda tener seguridad y un rápido acceso.
- Controlar la asignación y el uso de los recursos.
- Calendarizar los recursos de los usuarios.
- Facilitar el acceso a los dispositivos de Entrada/Salida.
- Recuperarse de fallas o errores.
- Manejar las comunicaciones a través de la red.
- Procurar la comunicación entre el hardware y el software.

En base a los puntos anteriores, debemos hacer un análisis muy detallado para lograr determinar cuales son los requerimientos más importantes de nuestra empresa y evaluar si el sistema operativo es capaz de satisfacer dichas demandas.

Bibliografía

Custer, Helen., *El Libro de Windows NT*, España, Mc Graw Hill, 1993.

E. Madnick, Stuart., J. Donovan, John., *Sistemas Operativos*, 2ª Edición, México, Diana, 1985.

Ezzell, Ben., Blaney, Jim., *NT/Windows Developer's Handbook*, Sybex, USA, 1997.

H. Rosen, Kenneth., R. Rosinski, Richard., M. Farber, James., A. Host, Douglas., *UNIX Sistema V Versión 4*, 2ª Edición, México, Mc Graw Hill, 1997.

H. Wood, Patrick., G. Kochan, Stephen., UNIX Networking, USA, Mc Millan, 1991.

Hahn, Harley., *UNIX Sin Fronteras*, México, Mc Graw Hill, 1995.

Lowell, Jay, Artur., *UNIX Shell Programming*, 2ª Edición, Canada, Mc Millan, 1990.

M. Deitel, Harvey., *Introducción a los Sistemas Operativos*, 2ª Edición EUA, Addison – Wesley, 1993.

Milenkovic, Milan., *Sistemas Operativos Conceptos y Diseño*, 2ª Edición, México, Mc Graw Hill, 1994.

Moritsugu, Steve., *Using UNIX*, 2ª Edición, USA, Mc Millan, 1998.

Nemeth, Evi., Snyder, Garth., Scott, Seebass., R. Hein, Trent., *UNIX System Administrator Handbook*, 2ª Edición, USA, Prentice Hall, 1995.

S. Tanenbaum, Andrew, *Sistemas Operativos Modernos*, México, Prentice Hall, 1993.

Silberschatz, Abraham., L. Peterson, James., B. Galvin, Peter., *Sistemas Operativos Fundamentales*, 3ª Edición, EUA, Addison –Wesley, 1994.

Singhal, Mukesh., G.Shivaratri, Niranjan., *Advanced Concepts in Operating Systems*, USA, Mc Graw Hill, 1994.

Stevens, W. Richard., *UNIX Network Programming Interprocess Communication*, 2ª Edición, USA, Prentice Hall, 1999.

Trackett, Jack., Gunter, David., *Linux Edición Especial*, 3ª Edición, España, Prentice Hall, 1998.

UNIX Systems Laboratories Inc., *UNIX System V Release 4 System Administrators Reference Manual*, USA, UNIX Press, 1990.

W. Kernighan, Brian., Pike Rob., *El Entorno de Programación UNIX*, México, Prentice Hall, 1987

Wayatt, Allen., *Windows NT Server*, Mexico, Prentice Hall, 1998.

Welsh, Matt., *Linux Instalation and First Steps*, 2ª Edición, USA, Lucas Project, 1999.