FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

A LOS ASISTENTES A LOS CURSOS

Las autoridades de la Facultad de Ingeniería, por conducto del jefe de la División de Educación Continua, otorgan una constancia de asistencia a quienes cumplan con los requisitos establecidos para cada curso.

El control de asistencia se llevará a cabo a través de la persona que le entregó las notas. Las inasistencias serán computadas por las autoridades de la División, con el fin de entregarle constancia solamente a los alumnos que tengan un mínimo de 80% de asistencias.

Pedimos a los asistentes recoger su constancia el día de la clausura. Estas se retendrán por el periodo de un año, pasado este tiempo la DECFI no se hará responsable de este documento.

Se recomienda a los asistentes participar activamente con sus ideas y experiencias, pues los cursos que ofrece la División están planeados para que los profesores expongan una tesis, pero sobre todo, para que coordinen las opiniones de todos los interesados, constituyendo verdaderos seminarios.

Es muy importante que todos los asistentes llenen y entreguen su hoja de inscripción al inicio del curso, información que servirá para integrar un directorio de asistentes, que se entregará oportunamente.

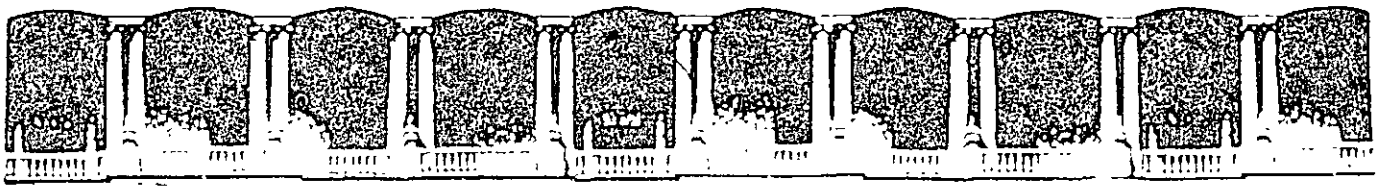
Con el objeto de mejorar los servicios que la División de Educación Continua ofrece, al final del curso deberán entregar la evaluación a través de un cuestionario diseñado para emitir juicios anónimos.

Se recomienda llenar dicha evaluación conforme los profesores impartan sus clases, a efecto de no llenar en la última sesión las evaluaciones y con esto sean más fehacientes sus apreciaciones.

Atentamente

División de Educación Continua.

Palacio de Minería Calle de Tacuba 5 Primer piso Deleg. Cuauhtémoc 06000 México, D.F. APDO. Postal M-2285
Teléfonos: 512-8955 512-5121 521-7335 521-1987 Fax 510-0573 521-4020 AL 26



FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

CURSOS INSTITUCIONALES
OFICIALÍA MAYOR DEL GOBIERNO DEL DISTRITO FEDERAL

UNIX BÁSICO

Del 18 al 22 de octubre de 1999.

Apuntes Generales

Ing. Saúl Magaña Cisneros
Ing. Juan Carlos Magaña Cisneros
Ing. Adrián Magaña Cisneros
Palacio de Minería
1999.

octubre de 1999

Temas

1 INTRODUCCIÓN

2 EDITOR VI

3 PRINCIPALES COMANDOS

4 ADMINISTRACIÓN

5 SHELL

6 UNIX INTERNET

INTRODUCCIÓN A UNIX

PRESENTACION

Si bien no es una novedad el **Sistema Operativo Unix** que nace en 1969 con su versión de Bell Laboratories, sí es saludable mencionar que en su evolución, ha logrado una superación de todos conocida en función de las necesidades informáticas. La importancia de la plataforma Unix se puede ilustrar con los datos estadísticos que nos mencionan que más del 70% de los servidores Internet se manejan bajo este sistema operativo.

Hoy en día, Unix domina en los ambientes de las macro y mini computadoras, en las "WorkStations" y tiene un crecimiento impresionante en redes de microcomputadoras corporativas. Esto último merced a su actual potencial logrado tras la evolución de las primeras versiones sobre plataformas Intel, e integrando los mundos **Windows-UNIX,**

Para la implementación de una red bajo al sistema operativo Unix será necesario el dominio de la configuración de los protocolos TCP/IP. La conformación de una red Unix, no sólo implica a este sistema operativo sino además la instalación de un sistema operativo de red bajo ésta plataforma, aspectos que son sustentados en este módulo.

Además se analizará el hardware especializado en el mundo Unix, como son tarjetas multipuertos, servidores de terminales, terminales X, y WorsStations

Se estudiarán también las diversas versiones de Unix encabezadas por SCO UNIX pasando por Solaris de Sun Microsystem, "AIX" de IBM, Hp-Ux , "Unixware" y Linux

Se mostrará el potencial de Unix en las comunicaciones, en el manejo de grandes bases de datos con la arquitectura Cliente-Servidor y en el manejo de aplicaciones de misión critica.

La DEC Conservando la vanguardia en la actualización profesional, ofrece este módulo como complemento del DIPLOMADO.

OBJETIVOS

Introducir a los participantes a UNIX y TPC/IP en ambientes de Red y hacerlos capaces de diseñar, instalar, administrar y dar mantenimiento a redes basadas en este sistema operativo y con esta familia de Protocolos.

A QUIEN VA DIRIGIDO

A profesionales, ejecutivos, y técnicos, que por sus necesidades profesionales y aplicaciones, requieran conocer instalar y administrar Redes sobre Unix y TCP/IP.

REQUISITOS.

Los participantes que estén con la modalidad de diplomado, deben haber cursado los módulos I y II.

Para los participantes en la modalidad de curso abierto, es necesario tener un buen nivel en microcomputación y conocimientos básicos de redes de computadoras así como el dominio de Windows 95, 98 con sus aplicaciones.

NOTA.- No es indispensable conocer Unix.

octubre de 1999

Temas

1 INTRODUCCIÓN

2 EDITOR VI

3 PRINCIPALES COMANDOS

4 ADMINISTRACION

5 SHELL

6 UNIX E INTERNET

INTRODUCCIÓN A UNIX

TEMARIO

1. INTRODUCCIÓN
- **Unix Y TCP/IP**
  - Introducción al sistema UNIX
  - Introducción a TCP/IP
- **UNIX en Red**
- **Fabricantes de Unix**

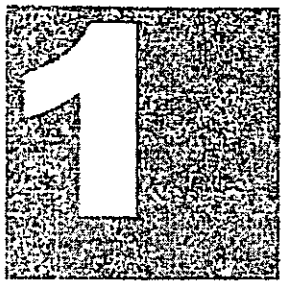
2-EDITOR VI
- Introducción
- Modos de Operación
- Descripción de Comandos

3 PRINCIPALES COMANDOS

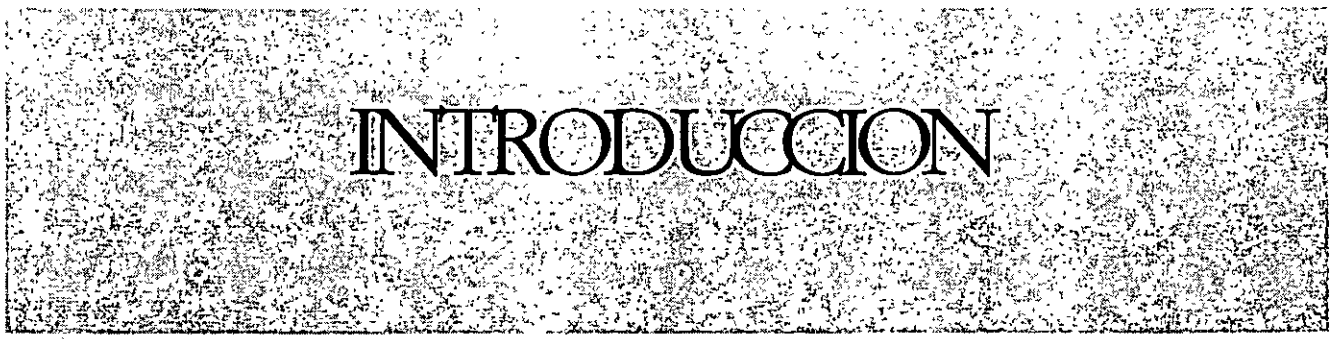
4. ADMINISTRACION
- **Usuarios**
- **Grupos de Usuarios**
- **Atributos y Seguridad**
- **Instalación de Aplicaciones**

5- SHELL
- Introducción
- Conectores (PIPES)
- Comodines

6- UNIX E INTERNET



INTRODUCCIÓN A UNIX



INTRODUCCION AL SISTEMA UNIX

HISTORIA DE UNIX

Hacia 1962, se emprendió el desarrollo de un sistema operativo multiusuario llamado MULTICS. La idea era tener una "planta de computación" enorme, como una planta eléctrica, para todo el mundo en Boston.

MULTICS llego a funcionar, pero fue demasiado ambicioso para la época. Contribuyo muchas ideas al mundo de sistemas operativos. De hecho, en Bell Labs, Ken Thompson, encontró una DEC PDP-7 (una de las primeras "minicomputadoras", del tamaño de un escritorio) y sobre esta plataforma desarrollo un sistema, que un poco en chiste se llamo "UNICS" por su contraste con el enorme MULTICS.

UNICS, cuyo nombre rápidamente paso a UNIX, fue traspasado de la PDP-7 a una PDP-11/20 y por fin a una PDP-11/20. En este momento, también en Bell Labs, Dennis Ritchie, colaboro con Thompson y juntos reescribieron UNIX en un lenguaje nuevo desarrollado por Ritchie, llamado C (porque venia de otro llamado B que a su vez venia de BCPL o Basic Computer Programming Language). De repente, UNIX (que de ahora en adelante escribiremos UNIX, como cualquier otro nombre propio) era portable a otras arquitecturas.

Una decisión histórica de AT&T causo que las universidades americanas pudieran obtener una licencia de UNIX, inclusive código fuente en C, por un precio irrisorio. La respuesta no se hizo esperar, y dentro de muy poco tiempo UNIX corría ("fue portado") a muchas otras arquitecturas. UNIX, de hecho, corre en mas arquitectura que cualquier otro sistema operativo.

Una de las universidades más influyentes en el desarrollo de Unix fue la Universidad de California en Berkeley (UCB). La UCB hizo muchas mejoras de Unix, inclusive en el sistema de archivos. Lamentablemente, esto origino también una fractura en las filas de Unix; ATT también siguió su desarrollo y los sistemas divergieron, creando casi dos Unixes. Hoy en día se habla de sistemas "convergidos" donde la convergencia se refiere a subsanar las diferencias entre el enfoque ATT y el enfoque de Berkeley.

Cuando surgen las estaciones de trabajo, debido a la iniciativa de SUN y Apollo en 1982 (y adelante), Unix estaba disponible y fue rápidamente portado a esta arquitectura, de modo que es el sistema operativo de rigor para las estaciones de trabajo.

A pesar de los sistemas abiertos, la historia de UNIX está dominada por el ascenso y caída de los sistemas hardware. UNIX nació en 1969 en una mainframe 635 de Genaral Electric. A la vez, los Laboratorios Bell de AT&T había completado el desarrollo de Multics, un sistema multiusuario que falló por su gran demanda de disco y

memoria. En respuesta a Multics, los ingenieros de sistemas Kenneth Thompson y Dennis Ritchie inventaron el UNIX. Inicialmente, Thompson y Ritchie diseñaron un sistema de archivos para su uso exclusivo, pero pronto lo cargaron en una Digital Equipment Corp. (DEC) PDP-7, una computadora con solo 18 kilobytes de memoria. Este suministraba una larga serie de puertos. En 1970, fue cargado en una PDP-11, y el runoff, el predecesor del troff, se convirtió en el primer procesador de texto de UNIX. En 1971, UNIX recibió reconocimiento oficial de AT&T cuando la firma lo usó para escribir manuales.

La segunda edición de UNIX fue realizada en 1971. La segunda edición dio forma al UNIX moderno con la introducción del lenguaje de programación C y sobre los 18 meses siguientes, el concepto de los pipes. Los pipes fueron importantes por muchas razones. Representaron una nueva forma de tratamiento de datos. Desde un punto de vista moderno, los pipes son un mecanismo orientado a objetos, porque entregan datos desde un objeto, o programa, a otro objeto.

Mientras tanto, había mucha actividad con el C. El C es otro producto de los Laboratorios Bell. Fue formado a partir de conceptos de otros tres lenguajes: B, CPL (Combined Programming Lenguage) y Algol-60. A finales de 1973, después de que Ritchie añadió soporte para variables globales y estructuras, C se convirtió en el lenguaje de programación de UNIX preferente. (Brian Kernigham, quien ayudó a Ritchie a desarrollar el C, añadió la R al estándar K&R, es el estándar preferido hasta la aceptación del ANSI C).

El ascenso del C fue responsable del concepto de portabilidad. Escrito en C, el entorno UNIX pudo ser relativamente fácil de trasladar a diferentes plataformas hardware. Las aplicaciones escritas en C pudieron ser fáciles de transportar entre diferentes variantes de UNIX. En esta situación nació el primer criterio de sistema abierto: portabilidad OS, la posibilidad de mover software desde una plataforma hardware a otra de una forma estándar. La portabilidad de UNIX se convirtió en el modelo de transportar aplicaciones en C desde un sistema UNIX a otro.

En 1974, la quinta edición de UNIX fue realizada para que estuviera disponible para las universidades. El precio de la versión 5 fue suficiente para recuperar los costos de las cintas y manuales. Se informó de los errores directamente a Thompson y Ritchie, quienes los reparaban a los pocos días de la notificación. En 1975, la sexta edición de UNIX fue desarrollada e iniciada para ser ampliamente usada. Durante este tiempo, los usuarios se hicieron activos, los grupos de usuarios fueron formados, y en 1976, se estableció el grupo de usuarios USENIX. En 1977, Interactive System Corp. inició la venta de UNIX en el mercado comercial. Durante este tiempo, UNIX también adquirió más poder, incluyendo soporte para procesadores punto flotante, microcódigo y administración de memoria.

Con la creciente popularidad de los microprocesadores, otras compañías trasladaron el UNIX a nuevas máquinas, pero su simplicidad y claridad tentó a muchos a aumentarlo bajo sus puntos de vista, resultando muchas variantes del sistema básico. En el período entre 1977 y 1982, los Laboratorios Bell combinaron algunas variantes de AT&T dentro de un sistema simple, conocido comercialmente como UNIX System III.

Los Laboratorios Bell más tarde añadieron muchas características nuevas al UNIX System III, llamando al nuevo producto UNIX System V, y AT&T anunció su apoyo oficial al System V en Enero de 1983. Sin embargo, algunas personas en la Universidad de California en Berkeley habían desarrollado una variante del UNIX, BSD, para máquinas VAX, incluyendo algunas nuevas e interesantes características.

A comienzos de 1984, había sobre 100.000 instalaciones del sistema UNIX en el mundo, funcionando en máquinas con un amplio rango de computadoras, desde microprocesadores hasta mainframes. Ningún otro sistema operativo puede hacer esta declaración. Muchas han sido las razones que han hecho posible la popularidad y el éxito del sistema UNIX:

- El sistema está escrito en un lenguaje de alto nivel, haciéndolo fácil de leer, comprender, cambiar, y mover a otras máquinas. Ritchie estimó que el primer sistema en C era de un 20 a un 40 por ciento más grande y lento porque no estaba escrito en lenguaje ensamblador, pero las ventajas de usar un lenguaje de alto nivel superaban largamente a las desventajas.
- Posee una simple interface de usuario con el poder de dar los servicios que los usuarios quieren.
- Provee de primitivas que permiten construir programas complejos a través de programas simples.
- Usa un sistema de archivos jerárquico que permite un mantenimiento fácil y una Implementación eficiente.
- Usa un formato consistente para los archivos, el flujo de bytes, haciendo a los programas de aplicación más fáciles de escribir.
- Provee una simple y consistente interface a los dispositivos periféricos.
- Es un sistema multiusuario y multitarèa; cada usuario puede ejecutar varios procesos simultáneamente.
- Oculta la arquitectura de la máquina al usuario, haciendo fácil de escribir programas que se ejecutan en diferentes implementaciones hardware.

Sin embargo tiene algunos inconvenientes:

- Comandos poco claros y con demasiadas opciones.
- Escasa protección entre usuarios.
- Sistema de archivo lento.

A pesar de que el sistema operativo y muchos de los comandos están escritos en C, UNIX soporta otros lenguajes, incluyendo Fortran, Basic, Pascal, Ada, Cobol, Lisp y Prolog. El sistema UNIX puede soportar cualquier lenguaje que tenga un compilador o intérprete y una interface de sistema que defina las peticiones del usuario de los servicios del sistema operativo de la forma estándar de las peticiones usadas en los sistemas UNIX.

Tabla 1. Hechos importantes en la historia de UNIX.

Año	Evento	Descripción

1965	Origen	Bell Telephone Laboratories y General Electric Company intervienen en el proyecto MAC (del MIT) para desarrollar MULTICS.
1969-71	Infancia del UNIX	El primer UNIX llamado Versión 1 o Primera edición, nace de las cenizas de MULTICS.
1972-73	Nace el C	En la Versión 2 el soporte del lenguaje C y los pipes son añadidos. En la Versión 4 el ciclo se completa con la reescritura de UNIX en C.
1974-75	El momento	Las Versiones 5 y 6 de UNIX se distribuyen a las universidades. La Versión 6 circula en algunos ambientes comerciales y gubernamentales. AT&T impone ahora pagar una licencia, a persar de que no puede promocionar UNIX por las duras regulaciones de EEUU del monopolio telefónico de AT&T.
1977	UNIX como producto	Interactive Systems es la primera compañía comercial que ofrece UNIX.
1977	Nace BSD	1BSD incluye un Shell Pascal, dispositivos y el editor ex.
1979	Versión 7	La Versión 7 de UNIX incluye el compilador completo K&R con uniones y definiciones de tipos. Versión 7 también añade el Bourne Shell.
1979	Trabajo en Red	BSD acrecentado por BBN incluye soporte para trabajar en red.
1979	Nace XENIX	Implementación para microcomputadoras ampliamente distribuido en hardware de bajo coste.
1980	Memoria Virtual	La capacidad de memoria virtual se añade en 4BSD.
1980	Nace ULTRIX	DEC realiza una versión de UNIX basado en BCD.
1980	Licencias en AT&T	La distribución de licencias abre el mercado.

1982	Atracción de los negocios	Soporte importante para procesos de transacciones desde UNIX System Development Lab.
1983	Nace System V	La versión más común de AT&T obtiene sus bases.
1984	Salida de SVR3	AT&T desata la versión más popular de System V hasta ahora.
1988	Motif vs Open Look	Sistemas por ventanas rivales son anunciados por OSF y UI.
1988	Siguiente paso	Un UNIX gráfico usa el **Kernell** Mach.
1990	OSF/1 vs SVR4	Versiones rivales de UNIX son anunciadas por OSF y UI.
1992-95	Socialización	OSF/1 abandona la escena; SVR4 se convierte en el estándar; Sun vende más estaciones de trabajo para usuarios de Motif que para usuarios de Open Windows; y crece Windows/NT de Microsoft.

VERSIONES DE UNIX

Cuando surgió UNIX su utilización fue limitada a sus creadores (los Laboratorios Bell) y a las instituciones tales como universidades y escuelas superiores, donde los usuarios tuvieron suficiente pericia y motivación para mantener un sistema operativo sin soporte por parte de los creadores. Esta situación y los atractivos obvios de su uso, llevaron de forma inevitable a cierto número de casas suministradoras de software a intentar que se tapara ese agujero y crear algún sistema operativo semejante a UNIX y accesible al gran mercado con el servicio de soporte que requiere un usuario comercial. Esto ha tenido dos tendencias:

- Comercialización de lo que es en esencia el sistema operativo UNIX, redirigido a otro procesador, debido al compilador transportable C, puesto que UNIX está escrito en lenguaje C.
- Comercialización de un UNIX reescrito, con igual apariencia pero métodos y estructuras internas de trabajo que pueden ser muy distintos.

La variedad entre las versiones de UNIX es terrible. Las dos familias más importantes de versiones de UNIX son BSD y System V. BSD dio nacimiento a SunOS, quien se ha convertido ahora en el progenitor de muchas pequeñas variantes en el mercado de las SPARC. Tatung, por ejemplo, ofertó SPARC-OS, y Solbourne Computers ofertó SolOS. Con la adquisición de la división de sistemas operativos de Interactive Systems, Sun ha trasladado también SunOS a las arquitecturas Intel x86.

System V es la versión más ampliamente usada de UNIX. Es el descendiente directo del UNIX desarrollado por AT&T en 1969. Está actualmente en la revisión 4.1 y a menudo es referenciado como SVR4, o System V Release 4. Ejemplos de descendientes de System V son ZEUS, XENIX (desarrollado por Microsoft), Idrix, **LINUX.**

Enlaces TCP/IP

TCP/IP (Transmission Control Protocol / Internet Protocol) es una familia de protocolos para interconectar computadoras de diversas naturalezas. Lo que se ha venido observando al paso de los años es que TCP/IP es un protocolo fuerte que no se ha visto desplazado por otros protocolos como se pensaba. Originalmente TCP/IP se creó por pedido del Pentágono y se usó en su principio para la red ARPA que interconectaba a varias universidades y centros de investigación relacionados con el Gobierno de los Estados Unidos.

Es interesante hacer notar que ARPA después derivó a ser Internet, la red más grande del mundo, Internet, que cuenta con millones de nodos.

La evolución de TCP/IP se remonta a los primeros años de la década de los 80 y según fué desarrollándose, se fué estandarizando.

La forma en que se desarrolla hoy en día, es por medio de un Comité llamado IAB, que está formado por personas altamente calificadas, así se publican trimestralmente las especificaciones de los protocolos o sus revisiones.

Existe una diferencia primordial en estos estándares y es que, para que un protocolo reciba el nombre de estándar, debe haberse probado exitosamente en redes reales durante varios meses, lo que garantiza la funcionalidad del mismo.

Desde su planeación, TCP/IP se pensó para ser independiente del medio físico de enlace, es esto precisamente lo que ha hecho que sea un protocolo ampliamente usado en enlaces de redes locales entre si, o bien, con redes amplias WAN.

Los ambientes que usan TCP/IP se basan en que cada elemento de la red tenga su dirección IP. El propósito de lo anterior es identificar de forma única a cada elemento del conjunto, para IP cada uno de los nodos de la red.

A los nodos que son computadoras se les denomina *hosts*, bajo la terminología de TCP/IP, y los Gateways son el equipo que tiene realmente funciones de ruteador, es importante notar que la connotación de estos términos bajo TCP/IP es diferente a la que normalmente nos hemos referido.

Las direcciones de IP tienen como objetivo:

1. Identificar de manera única cada nodo de una red o un grupo de redes.

2. Identificar también a miembros de la misma red.

3. Direccionar información entre un nodo y otro, aún cuando ambos estén en distintas redes.

4. Direccionar información a todos los miembros de una red o grupo de redes.

IP hace el trabajo de llevar y traer paquetes entre todas las redes que estén unidas y usando este protocolo, pero no nos garantiza que éstos lleguen a su destino. Para remediar esto, está TCP tampoco nos regula el flujo de paquetes.

TCP tiene funciones importantes, las que se mencionan a continuación:

1. - Secuenciamiento y reconocimiento de paquetes.
2. - Control del flujo de la información.

TCP partirá en paquetes la información y la enviará. A cada paquete se le asigna un número. El reconocimiento significa que cuando un nodo recibe varios paquetes, debe informar al que los está enviando que efectivamente los está recibiendo, de esta manera se logra un cierto control sobre la información que se está transmitiendo.

El hecho de poder enviar los paquetes significa que antes de poder establecer comunicación entre dos nodos, es necesario un *handshake* que es el momento en que el receptor y el transmisor se ponen de acuerdo para poder establecer la comunicación.

Existe una serie de tareas que TCP/IP realiza y que son de suma utilidad, tales como la emulación de terminales, para poder entrar a una diversidad de equipos, así como la transferencia de archivos entre computadoras.

Dentro de las aplicaciones cliente-servidor, una de las que mayor auge ha tenido ha sido la de bases de datos, teniendo por un lado el equipo corriendo al manejador de bases de datos, y por otro, a muchas PC's conectándose a él a través de diversas herramientas e interactuando con la información.
Es importante recordar que las aplicaciones que corren en las PC's se denominan clientes y el equipo que tiene la base de datos se denomina servidor o *motor* de base de datos.

Como se desea poder realizar esa conexión entre clientes y servidores no importando si éstos están en la misma red o en redes distantes, la solución más sencilla es que ambos: clientes y servidores, se comuniquen usando TCP/IP, de hecho es la forma en que se ha comercializado. Oracle, Sybase, Gupta, Informix y varios más, usan TCP/IP como su forma de transporte de datos y comandos entre clientes y servidores.

Terminología

Como en la mayoría de las disciplinas técnicas, en el terreno de las comunicaciones se cuenta también con un lenguaje propio.

Bytes y Octetos

En el medio de la computación es muy comúnmente utilizada la palabra *byte* para referirse a una cantidad de *8 bits*. Sin embargo, esta palabra también se utiliza para definir a la unidad más pequeña direccionable en una computadora. Una solución a este problema es el empleo de la palabra *octeto* para denotar una cantidad de *8 bits*.

Big Endians y Little Endians

La característica de almacenamiento de datos en una computadora se puede clasificar en dos ramas, *Big Endians* cuando la computadora almacena los datos de tal forma que siempre queda al inicio el *byte más significativo*; y *Little Endians* en el caso en que queda al principio el *byte menos significativo*.

Protocolos, Pilas y Conjuntos

Un *Protocolo* es un conjunto de reglas que gobiernan las acciones de comunicación.

Una *Pila de Protocolos* es un conjunto subdividido de protocolos que interactúan con el fin de proveer comunicación entre diversas aplicaciones.

Un *Conjunto de Protocolos* es una familia de protocolos que opera de manera conjunta a efecto de crear una plataforma consistente.

Host, Ruteador y Otros conceptos

Un *Host* es una computadora central que puede tener uno o más usuarios, un Host con capacidad de soporte a TCP/IP puede fungir como último punto de una comunicación.

Un *Ruteador* especifica los caminos que deben seguir los datos a través de una red. Anteriormente se adoptaba el término *Gateway* para definir lo que hoy se conoce comercialmente como *Ruteador,* término que hoy en día se emplea para hacer referencia a un sistema que efectúa cierta clase de traducción de protocolos.

Un *Nodo o Elemento de Red*, es toda aquella entidad en la red, sin importar si se trata de un Host, Ruteador o algún otro dispositivo.

NOMBRES Y DIRECCIONES

Nombres y Dominios

Tanto los nombres de la estructura de una Inter-Red como los de un sistema administrativo, son jerárquicos. Una Inter-Red está dividida en partes llamadas *Dominios*.

La responsabilidad de asignar nombres dentro de un dominio es tarea del administrador designado de ese dominio. Este administrador puede crear subdominios y delegar la autoridad de nombramiento a otro individuo de cada subdominio.

Ejemplos de Nombres de Inter-Red

Un nombre de Inter-Red puede describir a un sistema de una manera muy apropiada ya que su estructura se basa en la concatenación de etiquetas que hacen referencia a cada subdominio. El nombre de una Inter-Red puede ser escrito en mayúsculas o en minúsculas indistintamente:

TALLER.DIPLOM.DECFI.UNAM
unix.diplom.decfi.unam
Parte2.Diplom.Decfi.Unam
INTRO.DIPLOM.DECFI.UNAM

Es fácil entender la estructura jerárquica de estos nombres. Todas las divisiones de la Universidad se encuentran en el dominio UNAM de la Inter-Red. DECFI es el dominio de segundo nivel justo abajo del nivel UNAM. DIPLOM hace referencia a los diplomados impartidos por la DECFI de la UNAM y se encuentra como dominio de tercer nivel bajo DECFI. Finalmente el nombre del Host que identifica un sistema individual, inicia la cadena que define el nombre. Las partes adyacentes del nombre se separan por medio de puntos (.).

El tamaño límite de cada etiqueta es de 63 caracteres, pero el número máximo de caracteres por nombre es de 255 incluyendo los puntos separadores.

- Formatos de Direcciones

El IP utiliza direcciones para identificar a los Host y para enviarles información. Cada Host debe tener asignada una dirección IP que pueda utilizarse en comunicaciones reales. El nombre de un Host es traducido a su dirección IP mediante la tabla de relación de Nombres y Direcciones.
Una dirección IP es un valor binario de 32 bits que define el espacio total de direcciones que es un conjunto de numero de direcciones. El conjunto total de direcciones IP contiene 2^{32} números.

La notación *punto* es la forma más popular de expresar una dirección IP de tal forma que los usuarios finales pueden leerlas y escribirlas fácilmente. Cada octeto de las direcciones se convierte en un número decimal y cada número se separa por un punto (.). Por ejemplo, la dirección de TALLER.DIPLOM.DECFI.UNAM en notación de 32 bit binarios será:

10000010 10000100 00001011 00011111
130.132.11.31

Cabe hacer notar que el número más grande que puede aparecer en una notación separada por puntos es 255, que corresponde al número binario 11111111.

Una dirección IP se constituye de dos partes:

- Dirección de Red
- Dirección Local

La Dirección de Red identifica la Red a la cual está conectado ese nodo, la Dirección Local a su vez, identifica al nodo de manera individual.

- Direcciones Clase A, Clase B y Clase C

Las redes varían en tamaño. Existen tres formatos de direcciones diferentes para Inter-Redes que definen el uso dependiendo de su tamaño:

- Clase A para redes grandes
- Clase B para redes medianas
- Clase C para redes pequeñas.

Además de las clases A, B y C existen dos formatos de direcciones especiales, esto son: Clase D y Clase E. Los formatos de Clase D se utilizan para un *Multicasting* de IP que se emplea para distribuir un mensaje a un grupo de sistemas dispersos a través de la Inter-Red. La Clase E reserva su formato de direcciones para uso experimental exclusivamente.

Los primeros cuatro bits de cada dirección determinan su clase:

BITS INICIALES	CLASE
0xxx	A
10xx	B
110x	C
1110	D
1111	E

- Sub-Redes

Un administrador que desarrolla una implementación que cuenta con una dirección de Red Clase A o Clase B entiende la implicación de una complicada interconexión de Redes LAN y WAN. Es por eso que resulta práctico dividir en partes el espacio de direcciones de tal forma que corresponda a la estructura de la Red como una familia de Sub-Redes. Para llevar a cabo esto, es necesario descomponer la parte local de la dirección de la siguiente manera

Dirección de Red Dirección de Sub-Red Dirección de Host

La asignación de la dirección de Sub-Red frecuentemente se hace en un byte límite, un administrador que implementa direcciones Clase B como 156.33 debe utilizar su tercer byte para identificar las Sub-Redes, por ejemplo:

156.33.1
156.33.2
156.33.3

El cuarto byte será utilizado para identificar a los Hosts de manera individual dentro de una Sub-Red. Por otro lado, un administrador que implementa direcciones Clase C sólo tiene un espacio de dirección de un byte y deberá utilizar cuatro bits para las direcciones de los Host.

- Máscaras de Sub-Red

Una máscara de Sub-Red es una secuencia de 32 bits que cubre con unos (1s) las zonas correspondientes a la red y a la Sub-Red, y cubre con ceros (0s) la zona que le corresponde a la dirección del Host.

El tráfico de información se rutea hacia un Host, considerando las partes de Red y Sub-Red de su dirección IP. Es sencillo decir que tanto de una dirección corresponde a la dirección de red debido a los formatos estrictamente definidos para Clase A, Clase B y Clase C.

A efecto de reconocer cualquier tipo de campo, con un tamaño arbitrariamente elegido para la Sub-Red, se creó un parámetro de configuración denominado *Máscara de Sub-Red*. Consta de una secuencia de 32 bits. Los bits que incluyen a las direcciones de Red y de Sub-Red, se restablecen con 1.

Por ejemplo, un administrador de una Red Clase B con dirección 156.33 ha elegido hacer uso del tercer byte de todas las direcciones a fin de identificar las Sub-Redes, por lo tanto, la Mascara de Sub-Redes será:

11111111 11111111 11111111 00000000

La máscara de Sub-Red puede ser expresada de las siguientes maneras:
En notación de unos y ceros (1s y 0s):

1111111 11111111 11111111 00000000

Se puede expresar en notación *hexadecimal* como:

ffffff00
o alternativamente, en notación *punto* como:

255.255.255.0

Los Ruteadores que están conectados directamente a una Sub-Red se configuran con la máscara para la Sub-Red. Es común el uso de una sola máscara de Sub-Red a través de toda una Internet de una Corporación.

Si una Red contiene muchas líneas *punto a punto*, los números de Sub-Red se estarían desperdiciando debido a que sólo existen dos sistemas en cada Red *punto a punto*. El administrador debe optar por hacer uso de máscaras de 14 bits (255.255.255.255) para sus líneas *punto a punto*.

La máscara de Sub-Red para una red usualmente es sólo conocida por los ruteadores que se encuentran conectados directamente a la Red. Cuando se ejecutan protocolos de ruteo tradicionales, es imposible "ver desde afuera" de que manera se encuentra subdividida la Red.

Direcciones Especiales

Identificación de Redes

Es muy recomendable conocer la forma en que se debe utilizar la notación *punto* para la dirección de IP, a fin de hacer referencia a la Red. Por convención, esto se hace llenando con ceros la parte correspondiente a la dirección local de la dirección IP. Por ejemplo,

5.0.0.0 identifica una Red Clase A, 131,18,0,0 identifica a una Red Clase B y 201.49.16.0 identifica a una Red Clase C. La misma convención se sigue para la identificación de Sub-Redes con la desventaja de que nunca deben asignarse direcciones de este tipo a Host o a Ruteadores debido a que, por la notación empleada, es muy factible caer en una confusión.

Mensajes a Redes

La dirección de IP 255.255.255.255 tiene un propósito especial. Se emplea para enviar mensajes a todos los Host de la Red Local, aunque también es posible enviar un mensaje a cualquier Host de una Red Remota que se elija.

Esto se consigue llenando con unos (1) parte de Dirección Local de la Dirección de IP. Un mensaje se utiliza frecuentemente cuando un Host requiere la localización de un Servidor. Por ejemplo: suponiendo que un usuario desea enviar un mensaje a todos los nodos de una Red Ethernet Clase C con dirección 201.49.16.0, La dirección que deberá utilizar será:

201.49.16.255

El resultado de enviar un *datagrama de IP* en esta dirección será que dicho datagrama será turnado al ruteador que esté conectado a la red 201.49.16.0, entonces éste hará un *MAC layer broadcast* para entregar el mensaje a todos los Host de la Red. Es importante hacer notar que ningún Host debe tener asignada la dirección 201.49.16.255.

Mensajes a Sub-Redes

Un mensaje también puede ser enviado a una Sub-Red específica. Por ejemplo: Si la dirección 131.18.7.0 identifica a una Sub-Red de una Red Clase B, entonces la dirección que deberá emplearse para enviar un mensaje a todos los nodos de esta Sub-Red será 131.18.7.255.

La dirección 131.18.255.255 se puede seguir utilizando para enviar mensajes a todos los nodos de la Red Clase B completa. Los ruteadores de la configuración deberán ser lo suficientemente inteligentes para distribuir el mensaje enviado a cada Sub-Red. Si se le ha asignado el número 255 a alguna de las Sub-Redes se presentará un problema, debido a que no estará claro si el mensaje enviado en la dirección 131.18.255.255, iba dirigido a toda la Red Clase B, o únicamente a la Sub-Red 255. La única forma de evitar este tipo de percances es asignar a las Sub-Redes números diferentes de 255.

- *Direcciones de Regreso*

Así como existen mensajes que se envían a Redes o Sub-Redes específicas, también existen aquellas que nunca dejan el Host local. A efecto de hacer una prueba del software de Red, es muy útil contar con una dirección de regreso que define *"quien es el nodo emisor"*, mismo que funciona como receptor.

Para este efecto, se utiliza por convención cualquier dirección que comience con 127, por ejemplo:

127.0.0.1

Existen otros formatos de direcciones especiales que se emplean solo durante la inicialización del sistema. Estos formatos están reservados y no se pueden utilizar para identificar destinos. Por convención, la dirección 0.0.0.0 definirá a un Host específico de una Red específica, los demás Host de la misma red se definirán cambiando la parte que corresponde al Host en la dirección; por ejemplo: 0.0.0.5 identifica al Host 5 de una Red en específico.

- Domain Name System

A efecto de establecer una comunicación con un Host, es necesario conocer en que dirección se encuentra. Por lo regular, el usuario final conoce el nombre del Host con el que desea comunicarse, pero no así su dirección. En este caso ya sea el usuario final o la aplicación que éste haya invocado, tienen la necesidad de visualizar estas direcciones.

En Redes pequeñas y aisladas, se puede hacer frente a este problema teniendo una tabla central de mantenimiento en la que se establezca la relación nombre-dirección de Host, de esta forma, los Hosts individuales se mantendrán "al día" copiando esta tabla periódicamente.

El *Domain Name System (Sistema de Nombre del Dominio)* se implementó con el fin de brindar un mejor método para relacionar los nombres y direcciones en una Inter-Red. Los nombres y direcciones se guardan en *name servers* distribuidos a través de toda la Inter-Red.

Estos *name servers* se actualizan en forma local, así, la conexión, desconexión y/o el movimiento de un nodo se registra rápidamente y con precisión en un *primary authoritative server.* Debido a que la conversión nombre-dirección no es tan importante, la información es copiada a uno o más *secondary authoritative servers.*

Muchos proveedores ofrecen software que permiten una función de sistema como *name server.* Regularmente el software es una adaptación del Dominio de Inter-Red Berkeley *(BIND).* una corporación puede hacer uso de este software para ejecutar su servicio propio de *name servers* y opcionalmente, puede conectar su servicio de nombres al *Internet Domain Name System (Sistema de Nombres de Dominio de Inter-Red).*

Un producto capaz de llevar a cabo visualizaciones de DNS es una parte estándar de productos de TCP/IP y recibe el nombre de *resolver.*

- Address Resolution Protocol (ARP)

En una comunicación es necesario convertir los nombres de los nodos en sus direcciones de IP, antes de que la información pueda ser enviada de una estación a otra en una Red LAN, se debe llevar a cabo una segunda conversión ya que debe conocerse la dirección física del nodo destino. Para lograr esto se conocen. tres métodos:

- Configurar una tabla de valores directamente en cada nodo
- Configurar una tabla de valores en un servidor al cual puedan consultar los nodos.
- Conocer otros valores mediante el envío de una consulta en la Red LAN.

El ARP define un método basado en mensajes para una conversión dinámica entre direcciones de IP y direcciones físicas. ARP permite al administrador de la Red añadir nodos a una Red local o cambiar una interface de red de un nodo en especial, sin necesidad de actualizar manualmente las tablas de conversión de direcciones.

Los sistemas en la Red Local pueden hacer uso de ARP para encontrar información de las direcciones físicas para sí mismos. Cuando un Host desea establecer una comunicación con otro local, visualiza la dirección de IP de éste en su tabla de ARP. Si no encuentra esa dirección, el Host envía una petición ARP que contenga la dire**cción de IP destino.**

El Host destino reconoce su dirección de IP y lee la petición. Primeramente actualizará su propia tabla de conversión de direcciones con la dirección de IP y la dirección física del Host emisor. Entonces el Host receptor envía la dirección de su propia interface de red. Cuando el Host emisor recibe esta dirección, actualiza su tabla de ARP y queda listo para una nueva transmisión a través de la Red.

UNIX EN RED

CARACTERISTICAS EN AMBIENTE DE RED

El tipo de Red bajo Unix que empieza a parecer, consiste en enlaces principalmente Ethernet o Token Ring con equipos tipo Cliente basados principalmente en PC´s compactibles y equipos servidores que son Workstation o multiusuarios basados en Unix.

La razon de utilizar PC´s como cliente y no algun otro equipo consiste en la necesidad de muchos usuarios de poder correr las aplicaciones tradicionales de DOS ventanas tipo Microsoft que puede lanzar aplicaciones tanto de dos como de Unix. Las PC´s tambien requieren el servicio de archivos e impresoras: es decir, el usuario de la PC deberis acesar los discos e impresoras de los servidores Unix, como si estuvieran localmente conectados a la PC. Este servicio es parecido al que da un servidor en una red Novell. Con la capsacidad de lanzar aplicaiones de DOS, Unix caracter y Unix grafico (X11) desde cualquier servidor en la red y varias en distintas ventanas de la PC, la PC se vuelve un cliente universal.

La razon de utilizar servidores Unix dentro de las redes locales esta principalmente la posibilidad de correr aplicaciones de base de datos en estos servidores conjuntamente con las demas aplicaiones desarrolladas para Unix y ventanas (X11). Servidor Unix puede adicionalmente correr aplicaciones muy robustas los actuales provedores de bases de datos SQL, ORACLE, INFORMIX, SYB, INGRESS y PROGRESS, cuantan con versiones de sus productos para todas las plataformas Unix. con estas herramientas, los usuariuos pueden desarrollar con rapidez sus propias aplcaciones y correrlas en los equipos servidores de la red.
Los servicios que proporcionan los servidores Unix son los siguientes:

- Aplicaiones de base de datos (Principalmente SQL)
- Aplicaiones Unix.
- Servicio de archivos e impresoras para clientes DOS
- Aplicaiones "X" gráficas.
- Comunicaciones TCP/IP, X.25, monitoreo de la red, etc.

Las ventajas para los usuario de este tipo de red local son:

- Poder correr aplicaiones de dos simultaneamente con aplicaicones graficas y aplicaciones Unix de caracter.
- Coectividad a una red Ethernet (LAN, WAN y GAN), con TCP/IP se pueden conectar.
- Terminales tontas
- PC´s
- Terminales X
- Workstation
- Mainframes

- Heterogenidad de marcas. se pueden mezclar marcas diversas con SUN/HP/IBM/DEC, etc.
- Simetria. Los clientes pueden lanzar apliaciones de cualquier servidor. cualquier servidor puede ser también clientes.
- Sistemas abiertos.

Para lograr lo anterior se requiere la conjunción de diversas tecnologías, a pesar de lo complicado que puede resultar. Los beneficios son tan grandes que vale la pena el esfuerzo requerido para incorporar, ya que se convertirán en tecnologías de punta de los siguientes años.

Estaciones de Trabajo Workstation.

Durante los últimos años se ha visto un crecimiento muy fuerte en el uso de redes loxcales basadas en servidores Unís. Esta tendencia empezó con la introducion al merwcado de las poderosas Workstation (estaciones de trabajo) basadas en la tecnología RISC (computadoras con un conjunto de restricciones). Los atributos se estas worksatation hacen muy atractivo su uso como servidores en las redes locales, además de sus tradicional orientación de aplicaciones de graficas (CAD/CAM), desktop, publishing, CASE y diseño.

Las características que comparten las distintas marcas de estaciones de trabajo son las siguientes:

- procesador poderoso de 32 bits basado en tecnología RISC

Las workstations cuentan con un CPU con tecnología RISC que pueden proporcionar hasta 70 MIPS(millones de instrucciones por segundo) y con memorias centrales de 16 a 256 mb. Esta velocidad de proceso les permite correr aplicaciones de tipo gráfico (CAD-CAM), etc. O bien muchos procesos simultáneos en modo multiusuario.

- Pantalla grafica grande y ratón (mouse)

Todos los modelos Workstation cuentan con pantallas gráficas de 19" y generalmente de color. Las imágenes manejadas son bit-mapped, es decir que lo que se ve es la pantalla es un reflejo de bits en la memoria principal: al modificar este arreglo, automáticamente se cambia la imagen correspondiente. El mouse también permite mucha agilidad en la comunicación del usuario con el equipo. La posibilidad de crear ventanas, manipularlas y pasar imágenes de una ventana a otra son muy útiles cuando se está trabajando con varios procesos a la vez.

- Tarjetas Ethernet o Token Ring integradas.

Las Workstation se diseñaron para trabajar en red local. Tan es así que todos los modelos tienen integrada desde la fábrica la tarjeta Ethernet o Token Ring. El protocolo de comunicación más solicitado por las Workstation es TCP/IP y su gran ventaja es la diversidad de distintas computadoras que soportan. Desde una PC con DOS hasta mainframes se pueden conectar desde una misma red.

- Sistema Operativo Unís con Ventanas X11

Las distintas marcas de Workstation en el mercado tienen otro atributo que les dan cierta compactibilidad todas cuentan con el sistema operativo Unís, y el sistema de ventanas X11. Unís que originalmente se desarrolló en los Laboratorios Bell de AT&T, es un sistema operativo multitareas y multiusuario. Es robusto y se ha vuelto el standar para equipos multiusuario de tamaño mediano. A través del sistema de ventanas X11, diferentes modelos de Workstation pueden coexistir en la misma red local y compartir aplicaciones mutuamente. Con otro producto, NFS (network file system; sistema de archivos de la red), una Workstation en la red puede asociar el sistema de archivos de otra computadora y verlo como si fuera propio. Este atributo le permite vera ana red local como un solo sistema de cómputo.
Sistema de ventanas X11 y los GUI (interfaces gráficas del usuario). El sistema de ventanas X11 se desarrolló en el Instituto Tecnológico de Massachussets (M.T.I) a partir de 1985 y proviene de una sistema "W" de Windows. Las diez primeras versiones las realizaron tres personas del MIT, pero la versión 11 Tuvo apoyo de otras empresas como Digital Equipment, Hewlett Packard en IBM.
El paquete X11 consiste en una serie de subrutinas para el manejo y despliegue de imágenes con funciones para crearlas, expanderlas, moverlas, etc., y además de controlar las interrupciones de un dispositivo de apunte o mouse. Cuando se invoca el sistema de ventanas X, se arrancan dos programas: uno, llamado el servidor X, controla las imágenes en la pantalla y el otro es la aplicación en sí. Los dos programas pueden en la misma computadora o en dos diferentes, comunicándose a través de memoria o de la red local.
La gran ventaja de este sistema consiste en poder arrancar una aplicación en una computadora diferente y verla en sus propia pantalla. En este caso, la aplicación corre en la otra computadora mientras el servidor Z ésta corriendo en su propia computadora. El sistema es simétrico, es decir que, la otra computadora en la red también puede correr una aplicación en nuestra máquina y verla en la suya. También se pueden fabricar computadoras sencillas que consisten en una pantalla grande, una unidad de procesamiento simple y un teclado y mouse que corre el servidor X almacenado en un programa. Al conectarse a la red, estos dispositivos, llamados Terminales X, pueden correr aplicaciones en otras Workstation en la red y verlas en su pantalla. Existen programas también que pueden instalar en PC´s convirtiéndolas en terminales X.

- Grupos de Trabajo

Esta posibilidad de tener una aplicación corriendo en un equipo servidor X en otro, ha creado un nuevo concepto en la computación moderna, el grupo de trabajo. En este concepto, varias Workstation de distintas marcas pueden estar conectadas en una red local y pueden contar cada uno con distintas aplicaciones. Cualquier usuario de la red puede correr desde su equipo, cualquier aplicación que se encuentre en otro equipo, como si lo corriera en su propia computadora. Esto, aunado a la posibilidad de poder compartir la información guardada en los distintos discos, permite que diferentes personas conectadas a distintos equipos en la red utilicen una herramienta o aplicación en común, inclusive para un solo resultado final.

El concepto grupo de trabajo es el poder de trabajar, en conjunto, un grupo de personas conectadas en red con diferentes equipos de computo.

Fabricantes de Workstation

Los principales fabricantes de Workstation son Sun Microsystem, Hewlett Packard. Digital Equipment e IBM.

Sun Microsystem

Sun Microsystems es el fabricante más grande de Workstation con una participación del 38% del mercado. El procesador RISC que utiliza se llama SPARC y SUN ha intentado convertirlo es estándar en el mercado por medio de la ventana de las licencias de sus tecnología a otros fabricantes como Fujitsu y Tatung. Estos clonos de Sun entran a competir contra Sun y las demás Workstation para dar a la tecnología SPARC más penetración del mercado.

Hewlett Packard

HO adquirió a la empresa APOLLO, otro fabricante de Workstation y está uniendo la tecnología de ésta con la suya propia. Su línea de productos, Snake, está basada en un chip RISC propio llamado PA (precision Architecture). Actualmente este chip es una de los más rápidos en el mercado, superando a los 70 MIPS. HP también está buscando aliados en el uso de su chip con empresas como HITACHI. En1997, alcanzó el 20% del marcado.

Digital Equipment

DEC cuanta con una línea de Workstation llamada Decstation, basada en el chip procesador RISC de la empresa MIPS. Se formó una alianza de mas de 30 empresas denominadas ACE (Advance Computer Environment) para fabricar clonos usando la nueva versión de este chip MIPSB4000. Actualmente DEC liberó un chip "Alpha" de 64 bits con posibilidad de superar a los 200 MIPS. A raíz de esto DEC probablemente dejará el consorcio de ACE.

IBM

IBM entró tarde con un equipo RS-6000 basado en un chip RISC propietario llamado Power Architecture. IBM también ha hecho alianza con Apple Computer y Wang para expandir la venta de su tecnología. Actualmente cuanta con sólo 9% del marcado, pero esta participación va en aumento.

OTROS

Existen otros fabricantes de Workstation como Sequent Silicon, Graphics, CDC, etc. Cuya fracción del marcado es de 15%.

Es importante recalcar, que en la actualidad existe una verdadera guerra de precios estre todos estos fabricantes, con las consecuencias lógicas: baja de precios, aumento de la tecnología y poder de computo.
Una PC Intel también puede convertirse en una terminal X que corre un programa especial que emula el servidor X. La compañía AGE Logic produce un programa "Software for DOS" que hace esta función. Al utilizar el ambiente Windows de Microsoft, es posible que un usuario de una PC conectada un una red de Workstation pueda tener aplicaciones de DOS y correr simultáneamente con aplicaciones X; lo cual ofrece un ámbito verdaderamente poderoso y flexible.
Locus Computing Corporation es el comercializador independiente más grande del mundo en el desarrollo de aplicaciones basadas en la conectividad e interoperabilidad Unix-DOS.

Servicios

Locus ofrece al cliente servicios de desarrollo para fabricantes casa de Software, integradores de sistemas y usuarios finales.
El equipo personalizado de desarrollo de Locus directamente con fabricantes de arquitecturas para computadoras y sistemas operativos.
Por ejemplo, Locus fue el creador del sistema operativo AIX de IBM y en la actualidad diseña utilerías para el mismo.
Locus también ofrece al cliente un laboratorio en el cual cuanta con distintas plataformas, sistemas operativos para sus pruebas, además ofrece asesoría para una integración completa de su desarrollo.
Actualmente existe más de 500,000 instalaciones de Locus Computing Corporation en todo el mundo de los siguientes productos:

PC-INTERFACE
Es un software con caracteristicas de red el cual permite a usuarios con PC´s y/o Macintosh no cuenten con disco duro, el PCI podrá asignarse un disco virtual C.D. en el caso de contar con disco duro físico por medio del PCI se podrá contar con un disco D.
La transferencia de archivos entre discos virtuales; físicos será por medio de un copy en DOS.
Los recursos de impresión se hacen por medio del spooler de Unís y/o Xenix sin imortar que la aplicación esté en DOS.
Actualmente está liberada la versión 4.1 de PC-INTERFACE la cual ya contiene drivers (NOIS.DRV) la cual da soporte a Novell.

BENEFICIOS

- Requerimientos de memoria mínimos.
- Servidor UNÍS y/o Xenis no dedicado.
- Seguridad completa de información
- Capacidad para manejar mutiples sistemas Unís.
- Emulación de terminal VT220/VT100 para PC´s
- Emulación de terminal VT320/VT102 para Macintohs
- Ejecución de procesos remotos

- Ejecución de comandos Unís desde DOS y/o Mac.
- Soporta PC´s remotas
- Soporta MS Windows
- Soporta tarjetas Ethernet, Token Ring y puertos de comunicación RS-232

DRIVERS ETHERNET

- 3com 501, 502, 503, etc.
- Digital Equipment Corp. DEPCA, DE100, DE200.
- Excelan
- Recal Interlan
- Ungerman
- Western Digital WD8003 E.EB y EBI
- Western Digital WD8013 EBI
- Western Digital WD8003 E/A(MCA)
- Xircom Pocker Ethernet (paridad gemela no es soportada)
- NW 1000 y NW2000

DRIVERS TOKEN RING

- Tarjetas IBM (4 y 4/16)

SERVIDORES DE PC-INTERFACE INCLUIDOS EN VARIAS MARCAS DE UNÍS

- SCO Open Desktop
- AIX
- Interactive
- ATNT
- DELL

Existen 45distintas plataformas de PC-INTERFACE (servidor) y se cuentan con PC-INTERFACE para DOS con soporte a Windows y PC-INTERFACE para Macintosh.

TCP/IP para DOS

Es un producto de software el cual permite a computadoras personales basadas en DOS comunicarse con una gran variedad de servidores Unís y/o Xenix más comunes en la industria, permite establecer sesiones remotas desde la PC transferencia de archivos entres PC con las otras computadoras conectadas a la red.

Beneficios

- Integración completa de los protocolos TCP/IP, TCP, UDP, IP y ARP.
- Bajos requerimientos de memoria.
- Protocolos estandar FTP y TELNET.
  FTP (File Transporting Protocol; protocolo para el transporte de archivo)
  TELNET (proceso remotos incluye emulación de terminal VT220 e incluye modos de emulación H19, Vf53 y ANZY X.364)

- Aplicaciones de red distribuidas.
- Multifunciones de estaciones de trabajo.

  TCP/IP para DOS soporta a usuarios con programas de red utilizando una interface en programas de aplicaciones librerias socket con las que nosotros podemos desarrollar y modificar algunas librerías y utilerías incluidas en TCP/IP para DOS.

  Las tarjetas de comunicación soportadas son las mismas de la lista de PC-INTERFACE y en TCP/IP para DOS no esta soportado en el puerto de comunicaciones RS232.

Professional Workstations XP1000

workstations
solutions
graphics

buyer's guide

configure & buy
best fit
locate a reseller

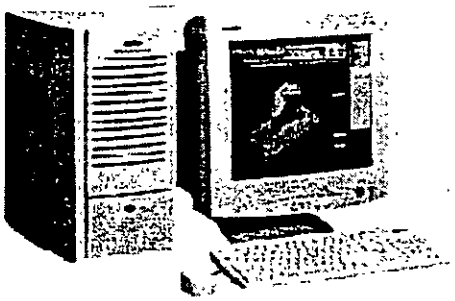
related links

specs & options
software platform
technical manuals
systems & options catalog
white papers
latest drivers
support

special offers

AlphaStation XP900
outstanding promotions

Unparalleled Performance and Operating System Flexibility



The Compaq Professional Workstation XP1000 is the capstone of the Professional Workstation family. This AlphaPowered™ workstation workstation incorporates the latest in Alpha processor technology and delivers award-winning performance on Compaq Tru64™ UNIX, Linux, and Windows NT. The XP1000 also offers support for high-performance 3D graphics, large memory capacity, fast system bus and other features high supporting performance computing.

CONFIGURE & BUY

Features

- Alpha 21264A 667 MHz or 21264 500 MHz processor
  - 64K/64K data/instruction cache
  - 4MB L2 cache
- Compaq Tru64 UNIX, Linux, and Windows NT Workstation 4.0 operating systems
- Industry leading SPECint95/fp95: 37.5/65.5
- 100 MHz Registered ECC SDRAM, expandable to 2 GB
- 9.1 GB or 18.2 GB Wide-Ultra SCSI (7,200 and 10,000 rpm)
- Graphics Options:
  - ELSA GLoria Synergy
  - Compaq PowerStorm 4D51T
  - PowerStorm 300 PCI and 350 PCI
- Minitower design that is optimized for use on the floor or is rack mountable

With the addition of the Extreme Performance line, Compaq offers the widest range of workstation platforms available from any vendor, anywhere.

1.800.345.1518

United States October 15, 1999

COMPAQ

AlphaStation™ XP900

workstations
solutions
graphics

buyer's guide

place an order

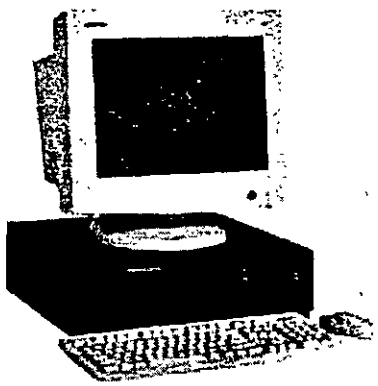
related links

Systems and Options Catalog

special interest

press release
OpenVMS Web site

Universal Platform, Low Cost Alpha Workstation



The AlphaStation XP900 is the new entry level Alpha 21264-based workstation targeted towards the technical user who demands the highest performance CPU at a low cost. The AlphaStation supports OpenVMS, Compaq Trü64™ UNIX, and Linux.

The AlphaStation XP900 brings you larger than life graphics with support for 3D, high end 2D, and multi-head graphics.

Features

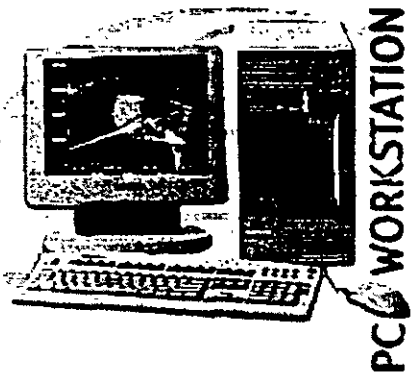
- High Performance system with Alpha 21264 466 MHz 64-bit processor
- Optimized to run UNIX, OpenVMS, and Linux.
- Up to 2GB ECC protected 100MHz SDRAM DIMM memory
- PowerStorm 350 or ELSA GLoria Synergy Graphics
- CD-ROM standard, plus room for 3 hard disk drives, or two hard disk drives and one tape drive
- Advanced high-bandwidth upgrade path for even more performance and reliability: single and dual channel UltraWide SCSI controllers, external RAID subsystems, and 10/100 Fast Ethernet controller.

1.800.345.1518

I-24

HP Kayak

Venezuela



■ La combinación precisa de poder de procesamiento y capacidad gráfica. Hewlett-Packard presenta su línea de PC Workstations HP Kayak, el más alto rendimiento para los ambientes de gran demanda.
Ya que cada segmento de profesionales requiere de una solución específica, Hewlett-Packard con sus tres modelos de HP Kayak le asegura estar trabajando con el equipo que se ajusta a sus necesidades de rendimiento y precio.

■ La HP Kayak XA PC Workstation está diseñada para soportar las necesidades de procesamiento de usuarios técnicos y de negocios con la necesidad de gráficos en 2D y 3D, manejo de modelos financieros, diseño de presentaciones y desarrollo de aplicaciones. Para los usuarios con mayores necesidades de procesamiento y capacidades de expansión HP posee la línea HP Kayak Xa-s.

El tope de la línea en PC Workstations, HP Kayak XU. Esta estación de trabajo de escritorio ha sido diseñada para cubrir las necesidades de los desarrolladores de aplicaciones, ingenieros, investigadores, analistas financieros y diseñadores creativos trabajando bajo MS Windows NT.

United States October 15, 1999

workstations
solutions
graphics

configure & buy
best fit
locate a reseller

specs & options
software platform
technical manuals
systems & options catalog
white papers
latest drivers
support

AlphaStation XP900
outstanding promotions

Unparalleled Performance and Operating System Flexibility

The Compaq Professional Workstation XP1000 is the capstone of the Professional Workstation family. This AlphaPowered™ workstation workstation incorporates the latest in Alpha processor technology and delivers award-winning performance on Compaq Tru64™ UNIX, Linux, and Windows NT. The XP1000 also offers support for high-performance 3D graphics, large memory capacity, fast system bus and other features high supporting performance computing.

Features

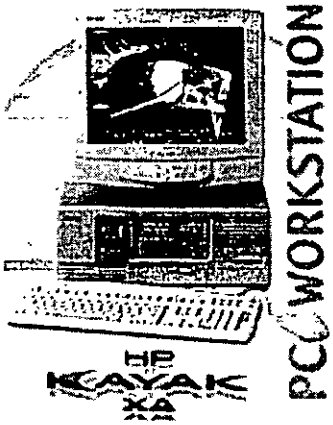
- Alpha 21264A 667 MHz or 21264 500 MHz processor
  - 64K/64K data/instruction cache
  - 4MB L2 cache
- Compaq Tru64 UNIX, Linux, and Windows NT Workstation 4.0 operating systems
- Industry leading SPECint95/fp95: 37.5/65.5
- 100 MHz Registered ECC SDRAM, expandable to 2 GB
- 9.1 GB or 18.2 GB Wide-Ultra SCSI (7,200 and 10,000 rpm)
- Graphics Options:
  - ELSA GLoria Synergy
  - Compaq PowerStorm 4D51T
  - PowerStorm 300 PCI and 350 PCI
- Minitower design that is optimized for use on the floor or is rack mountable

With the addition of the Extreme Performance line, Compaq offers the widest range of workstation platforms available from any vendor, anywhere.

privacy and legal statement

HP Kayak XA

Venezuela



- Las HP Kayak XA son las PC Workstations de nivel de entrada que realmente establecen el estándar de productividad y rendimiento en el campo de las estaciones de trabajo económicas. Proponen soluciones gráficas de alto rendimiento en 2D y 3D. Están diseñadas para satisfacer las necesidades de procesamiento de usuarios técnicos y de negocios.

- Procesador Intel Pentium II a 400MHz o 450MHz, o Intel Pentium III a 450MHz o 500MHz
- Cache L2 de 512KB. integrado al procesador
- AGPset 440BX de intel, FSB a 100MHz.
- Memoria Sincrónica Dinámica con detección y correción ultra rápida de errores a 100MHz.
- Disco Duro Ultra ATA o SCSI
- Avanzadas soluciones en gráficas AGP en 2D y 3D.
- MS Windows NT WorkStation 4.0 precargado
- CD ROM HP 32X
- Audio estéreo full duplex de alta fidelidad compatible con Sound Blaster Pro en todos los modelos
- Máxima confiabilidad y tiempo de operación con HP MaxiLife
- Mayor facilidad de uso y administración con HP TopTools y HP ConfigTailor (Herramienta de Instalación y recuperación de Software HP)

- ***Modelos y Configuración***

Nombre	*Procesador*	*Ram*	*Disco Duro*	*Red*	*#Producto*
HP Kayak XA Modelos de Escritorio	Pentium II 400 MHz	64 MB	4 3 GB UATA	Opcional	D6723N
	Pentium II 450 MHz	64 MB	6.4 GB UATA	Opcional	D6726N
	Pentium III 500 MHz	128 MB	10.1 GB UATA	Opcional	D6731N
HP Kayak XA Modelos Minitorre	Pentium III 450 MHz	128 MB	6 4 GB UATA	Opcional	D6730N
	Pentium III 500 MHz	128 MB	9.1 GB SCSI	10/100Base-T	D6732N

Ultra 5 & Ultra 10
Nuevos Anuncios

Ultra 10 Workstation

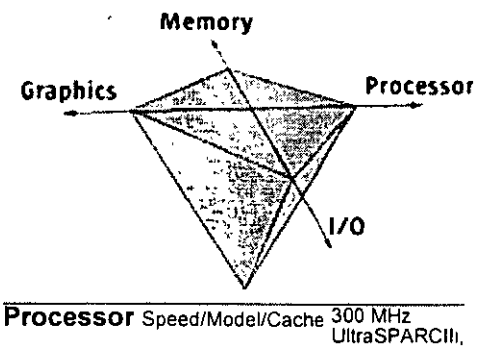
La Ultra10 resulta ideal para los usuarios de aplicaciones 3Dsensibles al precio, que trabajen en las áreas de:

- Diseño y realización de prototipos
- Animación, rendering y efectos de vídeo
- Geociencia (cartografía, prospecciones petrolíferas y de gas, etc.)
- Modelado y análisis
- Visualización científica

Ultra 10

Rendimiento global de aplicaciones más rápido. La Ultra 10 amplía la capacidad de la estación de trabajo Ultra 5 en todos los aspectos, asegurando una mejor ejecución global de sus aplicaciones, por lo que no sólo son más rápidas en unos cuantos benchmarks arbitrarios. Su arquitectura equilibrada significa que todos los elementos del sistema, procesador, memoria, E/S y gráficos operan a la máxima velocidad.

Sun's Balanced Architecture



Capaz de satisfacer sus requisitos inmediatos y futuros. La Ultra 10 fue diseñada para su ampliación al mismo ritmo que las necesidades de la informática técnica a medida que crezca su negocio.

		512kB
	integer benchmark	12.1 (SPECint95)
	floating point benchmark	12.9 (SPECfp95)
Memory	Size	1024 MB
	Throughput	600 MB/sec
I/O	Expansion Slots	4 32-bit 33MHz PCI
	Graphics Slot	1 UPA Slot, 100MHz
Graphics	Top system	Elite3D m3
	3D triangles	3 million triangles/sec
	3D benchmark	74 CDRS-03

Utiliza componentes estándar para que la expansión de memoria y periféricos resulte fácil y económica.

Opera transparentemente con estaciones de trabajo y PCs. La Ultra 10 presenta compatibilidad binaria al 100% con más de 12.000 aplicaciones originales Solaris, por lo que podrá elegir las mejores aplicaciones que se ajusten a sus necesidades. La interoperabilidad con los PCs facilita la ejecución de MS Office y otras aplicaciones de productividad, y, además, el Ultra 10 hace posible el intercambio de formatos.

Visualización e imágenes más rápidas y más realistas. La Ultra 10 tiene una amplia gama de capacidades gráficas que nunca se han ofrecido anteriormente en este rango de precios.

Servers
RS/6000 Home
Overview
Hardware
Software
How to Buy
Support
News
Solutions
Library
Site Map
RS/6000 Worldwide
Australia Go

RS/6000 43P Model 140

Related Links

- RS/6000 43P Model 140 specs

The 43P-140 workstation is designed for entry-level 2D to mid-range 3D graphics applications. Users can start simple and grow easily through a processor upgrade, additional memory, or graphics accelerators. The 43P-140 workstation can run entry-level 3D applications, such as production computer-aided design, at 2D system prices by using a 2D graphics accelerator in combination with Softgraphics 3D support in the OpenGL and graPHIGS application programming interfaces.

RS/6000 Model 43P-140 Workstation Comparison Chart

Machine type	**7043**	**7043**
Microprocessor		
Type	PowerPC 604e	PowerPC 604e
processors per system	1	1
Clock rates available	233 MHz	332 MHz
Memory		
System(min/max)	64 MB/768 MB	64 MB/768 MB
L2 Cache	1 MB	1 MB
Capacity		
Slots available	3 PCI + 2 PCI/ISA	3 PCI + 2 PCI/ISA
Disk/Media bays	5	5
Internal disk(min/max)	9 1 GB/54.6 GB	9.1 GB/54.6 GB
Benchmarks		
SPECint95	9 24*	12 9
SPECfp95	5.75*	6 21
POWER GXT2000P	—	---
PLBwire93/PLBsurf93	146.2/304.7	168/347
ProCDRS-02	---	7 2

***-may not be achievable with 43P-140 upgraded to 233MHz**

Request a quote | Buy now | ShopIBM/US

If you would like more information about this product or any of IBM's products and solutions, contact us.

Back to Workstations Overview

Servers

RS/6000 Home

Overview
Hardware
Software
How to Buy
Support
News
Solutions
Library
Site Map

RS/6000 Worldwide

Australia Go

RS/6000 43P Model 150

Related Links

- RS/6000 43P Model 150 specs

If you need to pump up your 3D graphics productivity while keeping a watchful eye on your budget, take a close look at the 43P-150 workstation. This compact yet expandable desktop system delivers the right combination of power, performance, and price to delight engineers and designers. A full range of graphics capabilities, both 2D and 3D, are available. Outstanding advanced 3D performance can be achieved at a modest cost using the POWER GXT2000P graphics accelerator.

RS/6000 Model 43P-150 Workstation Comparison Chart

Machine type	**7043**
Microprocessor	
Type	PowerPC 604e
processors per system	1
Clock rates available	375 MHz
Memory	
System(min/max)	128 MB/1 GB
L2 Cache	1 MB
Capacity	
Slots available	5 PCI
Disk/Media bays	5
Internal disk(min/max)	9.1 GB/54.6 GB
Benchmarks	
SPECint95	15.1
SPECfp95	10.1
POWER GXT2000P	
PLBwire93/PLBsurf93	257/464
ProCDRS-02	11 2

Request a quote

Buy now
ShopIBM/US

If you would like more information about this product or any of IBM's products and solutions, contact us.

Back to Workstations Overview

Servers
RS/6000 Home
Overview
Hardware
Software
How to Buy
Support
News
Solutions
Library
Site Map
RS/6000 Worldwide
Australia

RS/6000 43P Model 260

Related Links
- RS/6000 43P Model 260 specs
- MCAD
- White paper

The 43P-260 workstation is a 1- or 2-way symmetric multiprocessing workstation with breakthrough performance, flexibility, and affordability. Equipped with a choice of 3D graphics accelerators, it excels in performing the most sophisticated computer-assisted design, manufacturing, and engineering applications. The multithreaded 3D graphics application program interfaces available with AIX allow excellent scalability for graphics applications. As an entry-level 64-bit system, the 43P-260 workstation is also an excellent platform for application development and testing.

RS/6000 Model 43P-260 Workstation Comparison Chart

Machine type	**7043**	**7043**
Microprocessor		
Type	POWER3	POWER3
processors per system	1	2
Clock rates available	200 MHz	200 MHz
Memory		
System(min/max)	256 MB/8 GB*	256 MB/8 GB*
L2 Cache	4 MB**	4 MB**
Capacity		
Slots available	2 PCI (64-bit)+3 PCI (32-bit)	2 PCI (64-bit)+3 PCI (32-bit)
Disk/Media bays	5	5
Internal disk(min/max)	9 1 GB/54.6 GB	9.1 GB/54.6 GB
Benchmarks		
SPECint_base_rate95	104	205
SPECfp_base_rate95	225	434
POWER GXT3000P		
PLBwire93/PLBsurf93	426/674	714/898
ProCDRS-02	24.4	24.4

***-Shared memory**
****-per processor**

Request a quote Buy now
ShopIBM/US

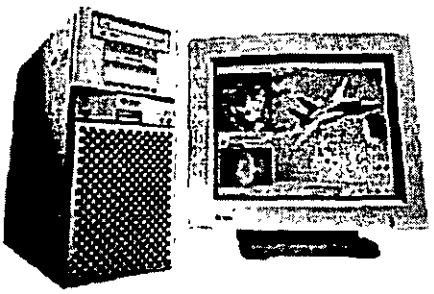
If you would like more information about this product or any of IBM's products and solutions, contact us.

Back to Workstations Overview

Estación de trabajo Ultra 60

La Ultra 60 es la estación de trabajo de sobremesa más rápida de Sun, tanto en configuraciones de un solo procesador como de dos procesadores, y resulta ideal para las siguientes actividades:

- Modelado y prototipos virtuales
- Animación, rendering y efectos de vídeo
- Geociencia (cartografía, prospecciones petrolíferas y de gas, etc.)
- Tratamiento y visualización de imágenes
- Tratamiento de imágenes médicas
- Investigación y desarrollo
- Diseño y análisis

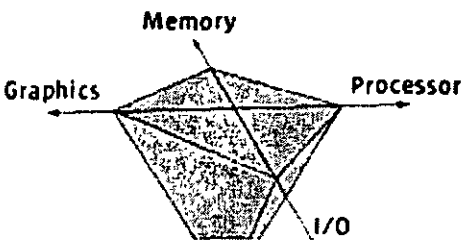


Ultra 60

Excelente rendimiento global de aplicaciones. La equilibrada arquitectura de la estación de trabajo Ultra 60 significa que puede implementar toda la potencia de proceso, memoria, E/S y gráficos que precise, sin limitaciones ni cuellos de botella en el rendimiento.

Capaz de satisfacer los requisitos actuales y futuros.

Sun's Balanced Architecture



La Ultra 60 tiene el ancho de banda de memoria más alto de la línea de Sun, lo que le permite llegar a los dos gigabytes de memoria, soportar dos procesadores y dos subsistemas gráficos basados en UPA, ejecutar varios arrays de discos desde sus dobles buses UltraSCSI, y ejecutar los periféricos de mayor velocidad en el bus PCI más rápido.

Opera con estaciones de trabajo y PCs. La Ultra 60 presenta compatibilidad binaria al 100% con todas las aplicaciones Solaris, lo que significa que podrá ejecutar las mismas aplicaciones de cualquier nivel de la línea de productos de Sun. La Ultra 60 interopera con PCs, por lo que se integra con el resto de la organización; además, el Ultra 60 hace posible el intercambio de medios.

Visualización y tratamiento de imágenes rápido y realista. La Ultra 60 permite trabajar con varios subsistemas gráficos de alto rendimiento, con Creator, Creator3D y Elite3D simultáneamente a la máxima velocidad, lo que lo convierte en el sistema ideal para los usuarios técnicos más exigentes.

Processor	Speed/Model/Cache	2 x 300 MHz UltraSPARCII, 2MB
	integer benchmark	13.1 (SPECint95)
	floating point benchmark	23.5 (SPECfp95)
Memory	Size	2048 MB
	Throughput	1.6 GB/sec, (1.9 GB/sec capable)
I/O	Expansion Slots	1 64-bit 66 MHz PCI 3 64/32-bit 33MHz PCI
	Throughput	400 MB/sec.
	Graphic Slots	2 UPA Slot, 100 MHz
	Throughput	800 MB/sec, (950 MB/sec capable)
Graphics	Top system	Elite3D m6
	2D benchmark	4.7 million 2D vectors/sec
	3D benchmark	125 CDRS-03

sgi™

Silicon Graphics® Visual Workstations

for Windows NT®

BUY ONLINE

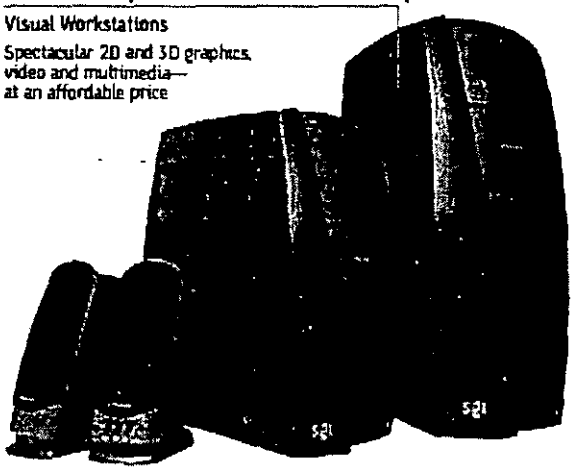
- Services & Support
- Company Info
- Help
- Resellers
- International Info
- Demos
- More SGI™ Products
- Site Map
- Contests

SEARCH GO

Silicon Graphics® 320 & Silicon Graphics® 540

Visual Workstations

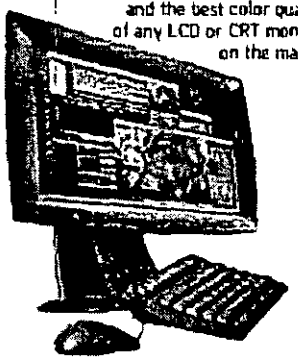
Spectacular 2D and 3D graphics, video and multimedia— at an affordable price



Silicon Graphics® 1600SW

Flat Panel Monitor

Ultra-fast pixel response and the best color quality of any LCD or CRT monitor on the market



Unleash the full potential of Windows NT.

Presenting Silicon Graphics® visual workstations for Windows NT. Featuring our Integrated Visual Computing (IVC) architecture with the Cobalt™ graphics chipset, they deliver unprecedented graphics and media performance with your Windows NT applications. Combined with the all-digital Silicon Graphics 1600SW flat-panel monitor, they're the ultimate tools for visual thinkers.

News: Silicon Graphics 320 with Pentium® III CPU at 600 MHz available to order...Support for Intel's new Pentium® III Xeon™ CPU at 550 MHz 1MB cache announced...LogiCad3D™ Magellan™ Plus 3D Motion Controller and Stereo Goggles Interface now available to order.

For U.S. customers only. Canadians please call (888) 400-4SGI

The **$c***parameter-count* command is a workaround.

sgi™

Silicon Graphics O2 Graphics

- **O2 Visual Workstation**
  - Home
  - Product Overview
  - System Hardware
  - Graphics
  - Digital Media
  - System Software
  - Technical Info
  - Promotions
  - Peripherals
  - Datasheets & White Papers
  - Service/Warranty
  - Demos/Tutorials
  - Related News
  - Contact Us
  - Awards
  - Events
- **Related Sites**
  - Octane Visual Workstations
  - Silicon Graphics 1600SW Flat Panel Monitor
  - Manufacturing
  - Entertainment
  - AEC

Technical Specs (pdf)

System Hardware

The Silicon Graphics® O2® visual workstation's industry-leading 3D graphics and image processing embody everything you've ever heard about SGI™ performance and realism. Built upon a native OpenGL® graphics subsystem and Unified Memory Architecture, the O2 system delivers a level of interactivity not available with any other machine in its class. These remarkable capabilities are standard with every configuration, giving you all the graphics and image-processing performance you need, right out of the box.

Advanced Graphics Features

With standard 32-bit double-buffered graphics and advanced features accelerated in hardware, O2 brings power and high quality within the reach of every user. Accelerated features include texture mapping, z-buffer, and anti-aliased points and lines, as well as stencil, fog, and color space conversions.

High-Performance Texture Mapping

The O2 workstations hardware-accelerated texture mapping capabilities bring a new level of realism and interactivity to the desktop. In CAD and animation, texture mapping allows users to visualize their models with a level of realism that surpasses traditional shaded models. Unlike traditional graphics boards that set a limit on texture memory, the flexible Unified Memory Architecture allows an unlimited amount of memory to be allocated for textures.

Consistent Feature Set across the SGI Product Line

Implementing key OpenGL hardware features of the higher-end Silicon Graphics Onyx2 InfiniteReality® systems, O2 systems now allows users to choose the level of graphics power at the price that best meets their needs.

First-Class Image Processing

The O2 systems deliver high-performance image-processing capabilities via hardware-accelerated OpenGL image-processing extensions and texture mapping. Implemented in hardware, these extensions allow users to manipulate large, high-resolution image data sets in real time, making it as easy to manipulate a 200MB image as a 2MB image. Delivering the most powerful performance in its class, O2 raises the standard of image-processing power in a desktop workstation.

Graphics Features

- 1280x1024 at 75Hz (also supports VGA, SVGA, XGA)
- Up to 32-bit RGBA double-buffered standard
- Native OpenGL graphics subsystem
- Hardware z-buffer (24-bit)
- Triangle rasterization in hardware
- Texture mapping in hardware
- Hardware image processing support
- Hardware stencil planes
- Hardware anti-aliasing
- Source plus destination alpha in hardware

O2 Graphics Libraries

OpenGL

OpenGL is an industry-standard graphics development environment. OpenGL is a cross-platform portable API that enables 2D, 3D, and imaging applications to be developed once, for deployment to a variety of hardware platforms with different operating systems and windowing environments. OpenGL includes operations for geometric and raster primitives, viewing and modeling transformations, lighting, shading, blending, fog, hidden surface removal, and texture mapping.

Open Inventor™

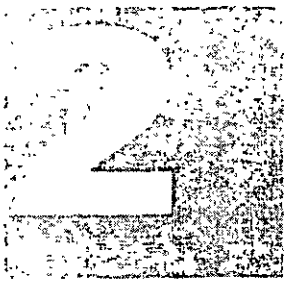
Open Inventor is an object-oriented 3D graphics toolkit that presents a programming model based on a 3D scene database. It provides a rich set of objects to speed up your development and extend 3D programming beyond OpenGL, X11, and Motif™. The objects include cubes, polygons, text, material, camera, lights, trackballs, handle boxes, and 3D viewers and editors. Like OpenGL, Open Inventor is a cross-platform portable API.

IRIS Performer™

The IRIS Performer toolkit gives developers the means to achieve a fast consistent frame rate for applications in the areas of visual simulation, interactive entertainment, and simulation-based design. IRIS Performer has the intelligence to evaluate underlying hardware and optimize for best performance. Key benefits include simplified management of the visual database and maximum performance for scene rendering for all Silicon Graphics workstations.

ImageVision Library®

The ImageVision Library is an object-oriented extensible toolkit for creating, processing, managing, and displaying images on all Silicon Graphics workstations. ImageVision Library was designed to reduce complexity for the development of image processing applications with an API that remains portable despite hardware or OS changes. The core set of robust image processing functions includes color conversion, arithmetic functions, radiometric and geometric transforms, and edge, line, and spot detection.



INTRODUCCIÓN A UNIX

EDITOR VI

EL EDITOR VI

INTRODUCCIÓN

El vi (visual) es un editor de textos eficaz (aunque críptico), interactivo y orientado visualmente. El vi aprovecha toda la pantalla del terminal para desplegar el texto que se está editando. Al usar VI, no es necesario hacer referencia a las líneas por sus números, puede ubicarse el cursor en forma manual en cualquier línea o carácter. El VI lleva un registro de lo que está en pantalla y la limpia sólo cuando es indispensable. Este manejo de la pantalla permite al 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.

El VI no es un programa de formato de texto. No justifica márgenes, ni centra títulos, ni tiene las características de un sistema de procesamiento de textos.

MODOS DE OPERACIÓN

VI es parte de otro editor llamado EX e implica a dos de los cinco modos de operación de EX, el modo de mandato y el modo de inserción. En el modo de mandato, VI acepta los tecleados como mandatos, y responde a todos los mandatos a medida que se introducen. En el modo de inserción, VI acepta como texto de tecleados, desplegando el texto conforme se introduce.

Al comienzo de una sesión de edición, VI se encuentra en el modo de mandato. Hay varios mandatos, como insertar y agregar, que colocan a VI en el modo de inserción. Cuando se presiona la tecla ESC, VI siempre regresa al modo de mandato.

Los mandatos cambiar y reemplazar combinan los modos de mandato y de inserción. El mandato cambiar borra el texto que se desea cambiar y coloca a VI en el modo de inserción para poder introducir texto nuevo. El mandato reemplazar borra el carácter o se sobreescribe e inserta el o los que se ingresan.

EDICIÓN

En esta sección se describe cómo llamar a VI, introducir texto y salir de VI. Todos los mandatos de VI son de efecto final inmediato; no es necesario oprimir RETURN para indicar el final de un mandato.
Cuando se le da VI un mandato, es importante distinguir entre letras mayúsculas o minúsculas.

LLAMADA A VI

Para crear en el directorio de trabajo un archivo denominado práctica se llama a VI con la línea de mandato siguiente.

$VI práctica

El archivo práctica es nuevo; todavía no tiene texto. VI despliega uno de los mensajes siguientes en la línea de estado (en la parte inferior) del terminal para indicar que se está creando y editando un archivo nuevo.

"práctica" No such file or directory.

o bien

"práctica" ERROR

Cuando se edita un archivo existente, VI despliega las primeras líneas del archivo y da información del estado de éste en la línea de estado.

INTRODUCCIÓN DE TEXTO

Colocación de VI en el modo de inserción. Una vez obtenido el acceso a VI. colóquese en el modo de inserción oprimiendo la tecla i. VI no emite ninguna señal para indicar que se encuentra en el modo de inserción.

Si no se tiene la seguridad de estar en el modo de inserción, presiónese la tecla ESC; VI regresará al modo de mandato si se encontraba en el modo de inserción o emitirá un aviso (un sonido agudo o una luz) si se encontraba ya en el modo de mandato. Puede regresar a VI al modo de inserción oprimiendo i de nuevo.

Introducción de texto. Mientras VI está en el modo de inserción, puede ponerse texto en el buffer de trabajo escribiendo en el terminal. Si el texto no aparece en la pantalla conforme se escribe, es porque no se está en el modo de inserción.

Introdúzcase el párrafo modelo que se muestra en la pantalla sig., presionando la tecla RETURN para terminar cada línea. Al introducir texto, hay que cuidar algunos detalles: impedir que las líneas de texto vuelvan del lado derecho de la pantalla, al izquierdo, oprimiendo la tecla RETURN antes de que el cursor llegue al final del extremo derecho. Hay que asegurarse también de no acabar una línea con un espacio, pues algunos mandatos VI se comportan en forma extraña cuando encuentran una línea que termina con un espacio.

modelo
VI (visual) es un editor de textos eficiente (aunque críptico), interactivo, orientado visualmente. VI aprovecha la pantalla completa del terminal desplegando el texto que se está editando.
~
~
~
Cuando se detecta un error en la línea que se está introduciendo, puede corregirse antes de continuar. Véase el párrafo siguiente. Más adelante pueden corregirse otros errores. Al terminar de introducir el párrafo, se oprime la tecla ESC para devolver VI al modo de mandato. La pantalla se verá como el modelo mostrado anteriormente.

Corrección de texto conforme se inserta. Las teclas que permiten retroceder y corregir una línea de mandato del Shell (por lo común CTRL-H, @ y #) realizan la misma función cuando VI se encuentra en el modo de inserción. Además, puede utilizarse CTRL-W para retroceder sobre palabras. VI puede no eliminar texto de la pantalla al retroceder sobre éste. Sin embargo, el texto es suprimido del buffer de trabajo.

Hay dos restricciones al uso de estas teclas de corrección. Sólo se toleran retroceder sobre texto en la línea que se está introduciendo (no se puede retroceder a una línea anterior) y sólo se harán sobre texto recién introducido. Como ejemplo, supongamos que se está en el modo de inserción introduciendo texto y se oprime la tecla ESC para devolver VI al modo de mandato. Ahora ya no es posible retroceder sobre el texto introducido la primera vez que se utilizó en el modo de inserción aunque el texto se encuentre en la línea actual.

TERMINACIÓN DE LA SESIÓN DE EDICIÓN

Puede concluirse la sesión de edición en una u otra de las formas siguientes: conservando los cambios realizados durante la sesión o sin conservarlos. En general se desea conservarlos.

Terminación normal. La terminación normal de una sesión de edición requiere que VI grabe el texto editado (el contenido del buffer de trabajo) antes de regresar el control al Shell. Esta forma de concluir una sesión de edición asegura que el archivo de disco refleje cualquier cambio realizado.

> Hay que asegurarse que VI se encuentra en el modo de mandatos y utilizar el mandato ZZ (deben ser mayúsculas) para escribir el texto recién introducido, desde el buffer de trabajo hasta el disco y terminar la sesión de edición. La única ocasión en que no debe usarse el mandato ZZ para concluir una sesión de edición es cuando no desea almacenar el texto editado.

Después de dar el mandato ZZ, VI despliega el nombre del archivo que se está editando y el número de caracteres en el archivo; después devuelve el control al Shell.

Terminación anormal. Algunas veces es necesario terminar una sesión de edición sin grabar el contenido del buffer de trabajo. Cuando se utiliza el mando :q! RETURN (el símbolo : mueve el cursor a la línea de estado) para concluir una sesión de edición, no se conserva nada del trabajo de la sesión de edición actual; el contenido del buffer de trabajo se pierde. La próxima vez que se edite o utilice el archivo, este aparecerá como era antes de empezar la sesión de edición actual. Este mandato ha de utilizarse con precaución.

MOVIMIENTO DEL CURSOR

Mientras VI está en el modo de mandato, puede colocarse el cursor encima de cualquier caracter de la pantalla. También pueden desplegarse en ésta distintas partes del buffer de trabajo. Manipulando la pantalla y la posición del cursor, éste puede situarse sobre cualquier caracter del buffer de trabajo.

movimiento del cursor por unidades de medida

MANDATO	MUEVE EL CURSOR
Espacio, 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 blancos a la derecha
B	una palabra a la izquierda
B	una palabra delimitada por blancos a la izquierda
$	fin de línea
O	principio de línea
RETURN	principio de siguiente línea
j o flecha descendente	hacia abajo una línea
K o flecha ascendente	hacia arriba una línea
)	fin de frase
(	principio de frase
}	fin de párrafo
{	principio de párrafo
}}	fin de archivo

MODO DE INSERCIÓN

Los mandatos de inserción, adición de texto, abrir líneas y reemplazar, colocan a VI en el modo de inserción. Mientras se encuentra en eso modo VI, puede ponerse texto nuevo en el buffer de trabajo. Al terminar de introducir texto, para devolver VI al modo de mandato, siempre se pulsa la tecla ESC.

El mandato de inserción

El mandato i coloca a VI en el modo de inserción y coloca el texto introducido antes del carácter sobre el cual se encuentra en el cursor (el caracter actual). Aunque el mandato i algunas veces escribe sobre el texto de la pantalla, éste reaparece el presionar ESC y devolver a VI al modo de mandato. Se utiliza el mandato i para insertar unos cuantos caracteres o palabras en un texto ya existente o para insertar texto en un nuevo archivo.

Los mandatos de adición (append)

El mandato a es similar al i, exepto en que pone el texto introducido después del carácter actual. El mandato A coloca el texto después del último caracter de la línea en curso.

Los mandatos de apertura (open)

Los mandatos o y O abren una línea en blanco dentro del texto existente, colocan el cursor al principio de la línea nueva (en blanco) y sitúan a VI en el modo de inserción. El mandato O abre una línea sobre la línea en curso; o la abre abajo. Se utilizan mandatos Open para introducir líneas nuevas en un texto ya existente.

Los mandatos de reemplázar (replace)

Los mandatos R y r hacen que el nuevo texto introducido sobreescriba (o reemplace) al existente. El caracter que sigue a un mandato r escribe sobre el caracter en curso. Después de ese caracter, VI regresa de forma automática al modo de mandato, sin necesidad de oprimir la tecla ESC.
El mandato R hace que todos los caracteres subsecuentes reemplacen el texto existente hasta pulsar ESC y devolver a VI al modo de mandato.

MODO DE MANDATO: BORRADO Y CAMBIO DE TEXTO

El mandato deshacer (undo)

El mandato deshacer, o u, deshace lo que acaba de hacerse. Restaura texto borrado o cambiado por error. El mandato Undo sólo arregla el último texto borrado. Si se borra una línea y después se cambia una palabra, el mandato sólo restaura la palabra cambiada, no la línea borrada. El mandato U restaura la línea actual a la forma en la que estaba antes de empezar a cambiarla, aunque se hayan realizado muchos cambios.

El mandato borrar un carácter (delete character)

El mandato x borra el caracter en curso. Si este mandato va seguido de un factor de repetición, entonces pueden borrarse varios caracteres de la línea actual, comenzando con el caracter actual.

El operador borrar (delete)

El operador d elimina texto del buffer de trabajo. La cantidad de texto que d suprime depende del factor de repetición y de la unidad de medida que se indican después de introducir d. Después de borrar el texto, VI se encuentra en el modo de mandato.
Advertencia: El mandato d RETURN, en forma ilógica, borra dos líneas, la línea en curso y la siguiente. Para borrar sólo la linea en curso se utiliza el mandato dd, o se antepone a dd un factor de repetición para borrar varias líneas.

BÚSQUEDA DE UNA CADENA

Los mandatos de búsqueda (search)

VI buscará por el buffer de trabajo una cadena de texto específica. Para encontrar la siguiente ocurrencia de una cadena (hacia adelante), oprímase la tecla diagonal (/),

digítese el texto que se desea localizar (llamado cadena de búsqueda) y presiónese RETURN. Al oprimir la tecla diagonal, se despliega una barra diagonal en la línea de estado y al introducir la cadena de texto, también esta aprecerá desplegada en la línea de estado. Cuando se oprime RETURN, VI busca la cadena, si la encuentra, coloca el cursor sobre el primer caracter de la cadena. Si se utiliza un signo de interrogación (?) en lugar de la barra diagonal, VI busca la existencia de una cadena anterior.

Las teclas N y n repiten la última búsqueda sin tener que introducir de nuevo la cadena de búsqueda. La tecla n repite de manera exacta la búsqueda original, mientras que N la repite en dirección opuesta.

La forma más cómoda de trabajar con un texto (modificarlo) es mediante un editor, Unix nos provee del editor VI, aunque su manejo es un tanto complejo, permite realizar operaciones que no todos los editores permiten.

La forma de invocarlo es:

% VI archivo

La mayoría de los comandos se dan pulsando las teclas indicadas sin que éstas aparezcan en la pantalla ni tampoco es necesario pulsar <ENTER> al final de ellos, sólo los comandos que comienzan con :,/ y ? son mostrados en la última línea de la pantalla y requieren la pulsación de <ENTER> para finalizar (estos corresponden a los comandos del editor EX, en el cual se basa VI).
Antes de comenzar a describir los comandos se establecerán las normas de la notación:

i	*texto*	< ESC >
(1)	(2)	(3)

1. Corresponde al comando, en este caso basta pulsar sólo la letra i
2. Aquí debe ingresar el texto que desee, puede utilizar más de una línea pulsando <ENTER> al final de cada una
3. Debe pulsar la tecla <ESC> (Escape)

c	Un caracter cualquiera
l	Una letra del alfabeto inglés
^X	Pulsar las teclas <CONTROL> y X
caracter	Un caracter cualquiera
CARACTER	Un caracter distinto de espacio

palabra	Una secuencia de letras y/o números
PALABRA	Una secuencia de caracteres incluyendo los espacios que siguen
arch	Algún archivo del disco (Existente o no)
patrón	Secuencia de caracteres a utilizar en un patrón de búsqueda
Movimiento	Algún comando de movimiento

DESCRIPCIÓN DE COMANDOS

A continuación se describirán la mayoría de los comandos de VI (sólo se excluyen los más complicados), a muchos de éstos se les puede anteponer un número decimal que indica un factor de repetición del comando, es decir, si se escribe 20 antes del comando, éste se repite 20 veces. Los comandos que tienen esta capacidad serán señalados con una letra n en la columna izquierda, al no colocar el valor, el comando se ejecuta sólo una vez.

COMANDOS DE MOVIMIENTO

Comando	Descripción
l ó	*n* caracteres a la derecha
h ó	*n* caracteres a la izquierda
k ó	*n* líneas arriba
j ó	*n* líneas abajo
^F	Avanza una página
^B	Retrocede una página
^D	Avanza media página
^U	Retrocede media página
^U	Retrocede media página
$	Avanza hasta el final de *n*-1 líneas adelante
^	Va al primer caracter distinto de espacio de la línea
_	Va al primer caracter distinto de espacio de *n*-1 líneas adelante
-	Va al primer caracter distinto de espacio *n* líneas atrás
+ ó <ENTER>	Va al primer caracter distinto de espacio *n* líneas adelante
0 (Cero)	Va al primer caracter de la línea (incluso espacio)
|	Va a la columna *n* dentro de la línea
f*c*	Avanza hasta el caracter *c*
t*c*	Avanza hasta la posición anterior al caracter *c*
F*c*	Retrocede hasta el caracter *c*
T*c*	Retrocede hasta la posición siguiente al caracter *c*
;	Repite el último comando 'f', 't', 'F' o 'T'
,	Idéntico al anterior, pero en la dirección opuesta
w	Avanza *n* palabras
W	Avanza *n* PALABRAS
b	Retrocede *n* palabras
B	Retrocede *n* PALABRAS
e	Avanza al final de *n* palabras adelante
E	Avanza al final de *n* PALABRAS adelante
G	Va a la línea *n* (última si *n* no se especifica)
H	Va a la línea *n* a partir de la primera que se ve en pantalla
L	Va a *n*-ésima línea anterior a la última que se ve en pantalla
M	Va a la línea del medio de la pantalla
)	Avanza *n* sentencias
(	Retrocede *n* sentencias

{	Avanza *n* párrafos
}	Retrocede *n* párrafos
`*l*	Va a la marca *l*
'*l*	Va al primer CARACTER dentro de la línea con la marca *l*
``	Va a la posición anterior al último salto
''	Va al primer CARACTER dentro de la línea en que se encontraba el cursor antes del último salto
/patrón	Avanza hasta la siguiente ocurrencia del *patrón*
?patrón	Retrocede a la anterior ocurrencia del *patrón*
n	Repite el último comando '/' o '?'
N	Igual al anterior, pero en la dirección opuesta
%	Busca el siguiente paréntesis o su pareja (también con {, }, [y])

COMANDOS DE INSERCIÓN DE TEXTO

Comando	Descripción
i*texto*<ESC>	Inserta *texto* en la posición actual del cursor
a*texto*<ESC>	Agrega *texto* en la posición siguiente a la del cursor
I*texto*<ESC>	Inserta *texto* delante del primer CARACTER de la línea actual
A*texto*<ESC>	Agrega *texto* al final de la línea actual
o*texto*<ESC>	Agrega *texto* en la línea siguiente
O*texto*<ESC>	Agrega *texto* en la línea anterior
p	Coloca el último grupo de líneas guardado o borrado en la línea siguiente *n* veces
P	Coloca el último grupo de líneas guardado o borrado en la línea anterior *n* veces
.	Repite el último comando *n* veces

COMANDOS DE REEMPLAZO

Comando	Descripción
r*c*	Reemplaza *n* caracteres por *c*
R*texto*<ESC>	Sobreescribe el resto de la línea, agregando *n*-1 veces
s	Sustituye *n* caracteres
S	Sustituye *n* líneas
c*movimiento* *texto*<ESC>	Cambia lo alcanzado por *n movimientos* por *texto*
cc*texto*<ESC>	Cambia *n* líneas por *texto*
C*texto*<ESC>	Cambia el resto de la línea las y *n*-1 líneas siguientes por *texto*
~	Intercambia entre mayúsculas y minúsculas
J	Junta *n* líneas (Si *n* no se especifica se junta la actual con la siguiente)
.	Repite el último comando *n* veces (J sólo una vez)
:[x,y]s/*patrón*/*texto*/m	Sustituye el texto alcanzado por *patrón* por *texto* entre las líneas x e y (% para señalar todas). El modificador *m* puede ser g (Global) o c (Con confirmación)
&	Repite el último reemplazo dado con el comando anterior

COMANDOS DE BORRADO

Comando	Descripción
x	Borra *n* caracteres a partir de la posición del cursor
X	Borra *n* caracteres antes del cursor
d*movimiento*	< Borra *n* veces lo indicado por *movimiento* (3dw' Borra 3 palabras)
dd	Borra *n* líneas
D	Borra hasta el final de la línea
.	Repite el último comando *n* veces

COMANDOS DE COPIA Y MARCADO

Comando	Descripción
y*movimiento*	Marca el texto descrito por *movimiento* para copiarlo con el comando p o P
yy	Marca *n* líneas para copiarlas con el comando p o P
Y	Marca *n* líneas para copiarlas con el comando p o P
m*l*	Marca la posición del cursor con la letra *l*

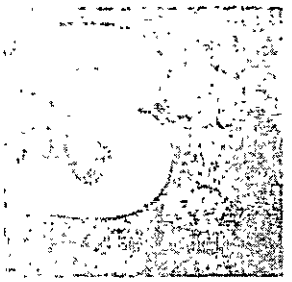
COMANDOS PARA DESHACER

Comando	**Descripción**
u	**Deshace la última modificación**
U	**Deshace todos los cambios hechos en la línea actual**
p	**Coloca el último grupo de líneas guardado o borrado en la línea siguiente**
P	Coloca el último grupo de líneas guardado o borrado en la línea anterior
:q!	Abandona VI sin grabar las modificaciones
:e!	Re-edita el archivo (Como salir y editarlo nuevamente)

COMANDOS DE GRABACIÓN Y SALIDA

Comando	Descripción
:q	Sale de VI (Si no se ha modificado desde la última grabación)
:q!	Sale sin grabar
:w	Graba el archivo
:w *arch*	Graba en el archivo *arch*
:w >> *arch*	Agrega el archivo editado al archivo de nombre *arch*
:w! *arch*	Graba el archivo editado con nombre *arch* sin importar que éste exista
:*x,y* w *arch*	Graba de la línea *x* a la *y* en el archivo *arch*
:wq	Graba y sale
:ZZ	Graba slo si el archivo ha sido modificado y sale

:f *arch*	Cambia el nombre del archivo editado a *arch*
:r *arch*	Agrega el archivo *arch* después de la línea actual



INTRODUCCIÓN A UNIX

PRINCIPALES COMANDOS

PRINCIPALES COMANDOS

COMANDO	DEFINICIÓN	SINTAXIS	EJEMPLO
cal	Despliega el calendario de un mes y año	cal <número_mes> <número_año>	cal 4 1999 /despliega el mes de abril de 1999
date	despliega la fecha actual	date	despliega "Mon Oct 18 13:42:38 1999
clear	Limpia pantalla	clear	
man	Despliega información sobre el comando tecleado	man <nombre_comando>	man date
cd	cambia de directorio	cd .. cd <nombre_directorio> cd /	Pasa al directorio superior Pasa dentro del directorio tecleado Pasa a la raíz del sistema
history	Lista los comandos previamente ejecutados	history <número_comandos>	history 10 /Lista los últimos diez comandos ejecutados
logout	Permite salirse de sesión	logout	
login	Permite identificarse ante el sistema	login <nombre_usuario>	login alumno
passwd	Cambia el password	passwd	
who	Lista los usuarios actualmente conectados al sistema	who	who am i /Indica con que nombre estoy identificado en el sistema
finger	Lista la información detallada acerca de los usuarios conectados	finger	

	actualmente al sistema.		
cat	Concatena y despliega archivos	cat <nombre_archivo>	cat examen
cp	Copia el contenido de archivos y directorios	cp <archivo1> <archivo2> cp <archivo> <ruta>	cp prueba prueba1 cp prueba /usuarios
mv	Mueva y renombra archivos	mv <archivo1> <archivo2> mv <archivo> <ruta>	mv fin final mv fin /usuarios
diff	Compara y despliega diferencias entre archivos	diff <archivo1> <archivo2>	diff fin finales /Despliega línea por línea las diferencias entre fin y finales
more	Despliega el contenido de un archivo en pantalla	more <nombre_archivo>	more final
cmp	Compara el contenido de dos archivos	cmp <archivo1> <archivo2>	cmp fin final
file	Despliega las características del archivo	file <nombre_archivo>	file fin
touch	Crea un archivo vacío	touch <nombre_archivo>	touch vacio
head	Despliega las primeras n líneas de un archivo	head -<número_lineas> <archivo>	head –10 prueba
tail	Despliega las últimas n líneas de un archivo	tail -<número_lineas> <archivo>	tail –5 prueba
sort	Ordena alfabéticamente líneas de un archivo y escribe la salida sobre la	sort –o <archivo_salida> <archivo_original>	sort –o ordena final sort –r final /ordena

	pantalla o un archivo	sort –r <nombre_archivo>	inversamente
rm	Borra un archivo	rm <nombre_archivo>	rm final
chmod	Permite modificar los modos de permisos de un archivo o directorio	chmod <clase> >operación> <permisos> <nombre_archivo>	chmod u = w permisos
COMANDOS	DEFINICIÓN	SINTAXIS	EJEMPLO
ln	Crea ligas entre archivos	ln <archivo1> <archivo2>	ln fin final /Crea una liga entre fin y final
pwd	Indica el directorio actual	pwd	
mkdir	Crea un directorio	mkdir <nombre_directorio>	mkdir curso
ls	Lista los archivos del directorio	ls ls –l	/Lista los nombres de los archivos del directorio /Lista los archivos, fecha,, hora, permisos, propietario, nombre del archivo
rmdir	Borra el directorio	rmdir <nombre_directorio>	rmdir directorio
echo	Despliega un mensaje en la pantalla	echo <Mensaje>	echo 'Hola'
grep	Busca texto o palabras en un archivo	grep <palabra_a_buscar> <archivo>	grep hola final
wc	Lista el número de líneas, palabras, caracteres en un archivo	wc –l wc –w wc –c	Cuenta Líneas Cuenta palabras Cuenta caracteres

which	Localiza un comando y muestra su ruta	which <comando>	which vi /despliega la ruta:/usr/ucb/vi
vi	Ejecuta el editor de texto del sistema	vi <archivo>	vi final
du	Da la cantidad de kilobytes usado en un directorio y todos sus subdirectorios o de un archivo	du –a <nombre_directorio> du –x <nombre_archivo>	Despliega la cantidad de kilobytes usada por un directorio y su contenido Despliega la cantidad de kilobytes de un archivo

while	**while**(1)	shell built-in functions to repetitively execute a set of actions while/until conditions are evaluated TRUE
who	**who**(1)	who is on the system
whoami	**whoami**(1B)	display the effective current username
whois	**whois**(1)	Internet user name directory service
write	**write**(1)	write to another user
xargs	**xargs**(1)	construct argument list(s) and execute command
xgettext	**xgettext**(1)	extract gettext call strings from C programs
xstr	**xstr**(1)	extract strings from C programs to implement shared strings
yacc	**yacc**(1)	yet another compiler-compiler
ypcat	**ypcat**(1)	print values in a NIS database
ypmatch	**ypmatch**(1)	print the value of one or more keys from a NIS map
yppasswd	**yppasswd**(1)	change your network password in the NIS database
ypwhich	**ypwhich**(1)	return name of NIS server or map master
zcat	**compress**(1)	compress, uncompress files or display expanded files

NAME

acctcom – search and print process accounting files

SYNOPSIS

acctcom [**-abfhikmqrtv**] [**-C** *sec*] [**-e** *time*] [**-E** *time*] [**-g** *group*] [**-H** *factor*] [**-I** *chars*] [**-l** *line*] [**-n** *pattern*] [**-o** *output-file*] [**-O** *sec*] [**-s** *time*] [**-S** *time*] [**-u** *user*] [*filename*...]

AVAILABILITY

SUNWaccu

DESCRIPTION

acctcom reads *filenames*, the standard input, or **/var/adm/pacct**, in the form described by **acct**(4) and writes selected records to standard output. Each record represents the execution of one process. The output shows the COMMAND NAME, USER, TTYNAME, START TIME, END TIME, REAL (SEC), CPU (SEC), MEAN SIZE (K), and optionally, F (the **fork()**/**exec()** flag: 1 for **fork()** without **exec()**), STAT (the system exit status), HOG FACTOR, KCORE MIN, CPU FACTOR, CHARS TRNSFD, and BLOCKS READ (total blocks read and written).

A ' # ' is prepended to the command name if the command was executed with super-user privileges. If a process is not associated with a known terminal, a ' ? ' is printed in the TTYNAME field.

If no *filename* is specified, and if the standard input is associated with a terminal or **/dev/null** (as is the case when using ' & ' in the shell), **/var/adm/pacct** is read; otherwise, the standard input is read.

If any *filename* arguments are given, they are read in their respective order. Each file is normally read forward, that is, in chronological order by process completion time. The file **/var/adm/pacct** is usually the current file to be examined; a busy system may need several such files of which all but the current file are found in **/var/adm/pacct***incr*.

OPTIONS

-a	Show some average statistics about the processes selected. The statistics will be printed after the output records.
-b	Read backwards, showing latest commands first. This option has no effect when standard input is read.
-f	Print the **fork()**/**exec()** flag and system exit status columns in the output. The numeric output for this option will be in octal.
-h	Instead of mean memory size, show the fraction of total available CPU time consumed by the process during its execution. This "hog factor" is computed as (total CPU time)/(elapsed time).
-i	Print columns containing the I/O counts in the output
-k	Instead of memory size, show total kcore-minutes.
-m	Show mean core size (the default).
-q	Do not print any output records, just print the average statistics as with the **-a** option.
-r	Show CPU factor (user-time/(system-time + user-time)).

-t	Show separate system and user CPU times.
-v	Exclude column headings from the output.
-C *sec*	Show only processes with total CPU time (system-time + user-time) exceeding *sec* seconds.
-e *time*	Select processes existing at or before *time*.
-E *time*	Select processes ending at or before *time*. Using the same *time* for both **-S** and **-E** shows the processes that existed at *time*.
-g *group*	Show only processes belonging to *group*. The *group* may be designated by either the group ID or group name.
-H *factor*	Show only processes that exceed *factor*, where factor is the "hog factor" as explained in option **-h** above.
-I *chars*	Show only processes transferring more characters than the cutoff number given by *chars*.
-l *line*	Show only processes belonging to terminal **/dev/term/***line*.
-n *pattern*	Show only commands matching *pattern* that may be a regular expression as in **regcmp**(3G), except + means one or more occurrences.
-o *output-file*	Copy selected process records in the input data format to *output-file*; suppress printing to standard output.
-O *sec*	Show only processes with CPU system time exceeding *sec* seconds.
-s *time*	Select processes existing at or after *time*, given in the format *hr* [:*min* [:*sec*]].
-S *time*	Select processes starting at or after *time*.
-u *user*	Show only processes belonging to *user*. The user may be specified by a user ID, a login name that is then converted to a user ID, ' # ' (which designates only those processes executed with superuser privileges), or ' ? ' (which designates only those processes associated with unknown user IDs).

FILES

/etc/group
/etc/passwd
/var/adm/pacct*incr*

SEE ALSO

ps(1), **acct**(1M), **acctcms**(1M), **acctcon**(1M), **acctmerg**(1M), **acctprc**(1M), **acctsh**(1M), **fwtmp**(1M), **runacct**(1M), **su**(1M), **acct**(2), **regcmp**(3G), **acct**(4), **utmp**(4)

Security, Performance, and Accounting Administration

NOTES

acctcom reports only on processes that have terminated; use **ps**(1) for active processes.

If *time* exceeds the present time, then *time* is interpreted as occurring on the previous day.

NAME

adb – general-purpose debugger

SYNOPSIS

adb [**–w**] [**–k**] [**–I** *dir*] [**–P** *prompt*] [*objectfile* [*corefile* [*swapfile*]]]

AVAILABILITY

SUNWtoo

DESCRIPTION

adb is an interactive, general-purpose debugger. It can be used to examine files and provides a controlled environment for the execution of programs.

objectfile is normally an executable program file, preferably containing a symbol table. If the file does not contain a symbol table, it can still be examined, but the symbolic features of **adb** cannot be used. The default for *objectfile* is **a.out**. *corefile* is assumed to be a core image file produced after executing *objectfile*. The default for *corefile* is **core**. *swapfile* is the image of the swap device used. It is valid only when used with the **–k** option.

OPTIONS

–w	Create both *objectfile* and *corefile*, if necessary, and open them for reading and writing so that they can be modified using **adb**.
–k	Perform kernel memory mapping; use when *corefile* is a system crash dump or **/dev/mem**, or when using a *swapfile*.
–I *dir*	Specify a directory where files to be read with **$<** or **$<<** (see below) will be sought; the default is **/usr/kvm/lib/adb**.
–P *prompt*	Specify the **adb** prompt string.

USAGE

adb reads commands from the standard input and displays responses on the standard output. It does not supply a prompt by default. It ignores the QUIT signal. INTERRUPT invokes the next **adb** command. **adb** generally recognizes command input of the form:

[*address*] [, *count*] [*command*] [;]

address and *count* (if supplied) are expressions that result, respectively, in a new current address, and a repetition count. *command* is composed of a verb followed by a modifier or list of modifiers.

The symbol '.' represents the current location It is initially zero. The default *count* is '1'.

Expressions

.	The value of *dot*.
+	The value of *dot* incremented by the current increment.
^	The value of *dot* decremented by the current increment
&	The last *address* typed. (In older versions of **adb**, "" was used.)
integer	A number. The prefixes **0o** and **0O** indicate octal; **0t** and **0T**, decimal; **0x** and **0X**, hexadecimal (the default).
int.frac	A floating-point number.
'cccc'	ASCII value of up to 4 characters.
<name	The value of *name*, which is either a variable name or a register name.
symbol	A symbol in the symbol table.
(exp)	The value of *exp*.

Unary Operators

**exp*	The contents of location *exp* in *corefile*.
%exp	The contents of location *exp* in *objectfile* (In older versions of adb, '@' was used).
–exp	Integer negation.
~exp	Bitwise complement.
#exp	Logical negation.

Binary Operators

Binary operators are left associative and have lower precedence than unary operators.

+	Integer addition.
–	Integer subtraction.
*	Integer multiplication.
%	Integer division.
&	Bitwise conjunction ("AND").
|	Bitwise disjunction ("OR").
#	*lhs* rounded up to the next multiple of *rhs*.

Variables

Named variables are set initially by **adb** but are not used subsequently.

0	The last value printed.
1	The last offset part of an instruction source.
2	The previous value of variable 1.
9	The count on the last **$<** or **$<<** command.

On entry the following are set from the system header in the *corefile* or *objectfile* as appropriate.

b	The base address of the data segment.
d	The data segment size.
e	The entry point.
m	The 'magic' number
t	The text segment size.

Commands

Commands to **adb** consist of a *verb* followed by a *modifier* or list of modifiers.

Verbs

?	Print locations starting at *address* in *objectfile*.
/	Print locations starting at *address* in *corefile*.
=	Print the value of *address* itself.
:	Manage a subprocess.
>	Assign a value to a variable or register.
RETURN	Repeat the previous command with a *count* of 1. Increment '.'.
!	Shell escape.

?, /, and = Modifiers

The following format modifiers apply to the commands ?, /, and = To specify a format, follow the command with an optional repeat count, and the desired format letter or letters:

{ ?,/,= } [[*r*] *f*...]]

where *r* is a repeat count, and *f* is one of the format letters listed below:

o	('.' increment: 2) Print 2 bytes in octal.
O	(4) Print 4 bytes in octal.
q	(2) Print in signed octal.
Q	(4) Print long signed octal.
d	(2) Print in decimal.
D	(4) Print long decimal.
x	(2) Print 2 bytes in hexadecimal.
X	(4) Print 4 bytes in hexadecimal.
u	(2) Print as an unsigned decimal number.
U	(4) Print long unsigned decimal.
f	(4) Print a single-precision floating-point number.
F	(8) Print a double-precision floating-point number.
b	(1) Print the addressed byte in octal.
c	(1) Print the addressed character.
C	(1) Print the addressed character using ^ escape convention
s	(*n*) Print the addressed string.
S	(*n*) Print a string using the ^ escape convention.
Y	(4) Print 4 bytes in date format
i	(*4* on SPARC; *n* on x86) Print as machine instructions.
a	(0) Print the value of '.' in symbolic form.
p	(4) Print the addressed value in symbolic form.
t	(0) Tab to the next appropriate TAB stop.
r	(0) Print a SPACE.
n	(0) Print a NEWLINE.
"..."	(0) Print the enclosed string.
^	(0) Decrement '.'.
+	(0) Increment '.'.
–	(0) Decrement '.' by 1.

? and / Modifiers

l *value mask*	Apply *mask* and compare for *value*; move '.' to matching location.
L *value mask*	Apply *mask* and compare for 4-byte *value*; move '.' to matching location.
w *value*	Write the 2-byte *value* to address.
W *value*	Write the 4-byte *value* to address.
m *b1 e1 f1*[?]	Map new values for *b1, e1, f1*. If the ? or / is followed by * then the second segment (*b2, e2, f2*) of the address mapping is changed.
v	Like w, but writes only bytes at a time.

: *Modifiers*	**b** *commands*	Set breakpoint, execute *commands* when reached.
	r	Run *objectfile* as a subprocess.
	d	Delete breakpoint at *address*.
	z	Delete all breakpoints.
	c*s*	x86: The subprocess is continued with signal *s*.
	s*s*	Single-step the subprocess with signal *s*.
	i	Add the signal specified by *address* to the list of signals passed directly to the subprocess.
	t	Remove the signal specified by *address* from the list implicitly passed to the subprocess.
	k	Terminate (kill) the current subprocess, if any.
	A	Attach **adb** to an existing process id. (For example, **0t1234:A** would attach **adb** to decimal process number **1234**.)
	R	Release the previously attached process.
$ *Modifiers*	<*filename*	Read commands from the file *filename*.
	<<*filename*	Similar to <, but can be used in a file of commands without closing the file.
	>*filename*	Append output to *filename*, which is created if it does not exist.
	l	x86: Show the current lightweight process (lwp) ID.
	L	x86: Show all the lwp IDs.
	P	Specify the **adb** prompt string.
	?	Print process ID, the signal which stopped the subprocess, and the registers.
	r	Print the names and contents of the general CPU registers, and the instruction addressed by **pc**.
	x or **X**	x86: Print the contents of floating point registers. **$x** and **$X** accept a "count" which determines the precision in which the floating point registers will be printed; the default is **25**. Using **$X** will produce more verbose output than using **$x**.
	x	SPARC: Print the names and contents of floating-point registers 0 through 15.
	X	SPARC: Print the names and contents of floating-point registers 16 through 31.
	b	Print all breakpoints and their associated counts and commands.
	c	C stack backtrace. On Sun-4 systems, it is impossible for **adb** to determine how many parameters were passed to a function. The default that **adb** chooses in a **$c** command is to show the six parameter registers. This can be overridden by appending a hexadecimal number to the **$c** command, specifying how many parameters to display. For example, the **$cf** command will print 15 parameters for each function in the stack trace.
	C	x86: Same as **$c**, but in addition it displays the frame pointer values.
	d	Set the default radix to *address* and report the new value. Note: *address* is interpreted in the (old) current radix. Thus '**10$d**' never changes the default radix.

e	Print the names and values of external variables.
w	Set the page width for output to *address* (default 80).
s	Set the limit for symbol matches to *address* (default 255).
o	All integers input are regarded as octal.
q	Exit from **adb**.
v	Print all non zero variables in octal.
m	Print the address map.
f	Print a list of known source filenames.
p	(*Kernel debugging*) Change the current kernel memory mapping to map the designated **user structure** to the address given by *u* ; this is the address of the user's **proc** structure.
i	Show which signals are passed to the subprocess with the minimum of **adb** interference.
W	Reopen *objectfile* and *corefile* for writing, as though the **–w** command-line argument had been given.

EXAMPLES

To start **adb** on the running kernel, use (as **root**):

```
example# adb -k /dev/ksyms /dev/mem
```

/dev/ksyms is a special driver that provides an image of the kernel's symbol table. This can be used to examine kernel state and debug device drivers. Refer to the Debugging chapter in *Writing Device Drivers* for more information.

FILES

/usr/kvm/lib/adb	default directory in which files are to be read with **$<** and **$<<**.
a.out	
core	
/dev/ksyms	special driver to provide an image of the kernel's symbolic table.

SEE ALSO

ptrace(2), **a.out**(4), **core**(4), **proc**(4), **ksyms**(7)

Writing Device Drivers

DIAGNOSTICS

adb, when there is no current command or format, comments about inaccessible files, syntax errors, abnormal termination of commands, etc. Exit status is **0**, unless last command failed or returned nonzero status.

NOTES

adb should be changed to use the new format symbolic information generated by **–g**.

adb is platform and release dependent. Kernel core dumps should be examined on the same platform they were created on.

BUGS

On SPARC systems, there does not seem to be any way to clear all breakpoints.

Since no shell is invoked to interpret the arguments of the **:r** command, the customary wild-card and variable expansions cannot occur.

Since there is little type-checking on addresses, using a sourcefile address in an inappropriate context may lead to unexpected results.

NAME

addbib – create or extend a bibliographic database

SYNOPSIS

addbib [**–a**] [**–p** *promptfile*] *database*

AVAILABILITY

SUNWdoc

DESCRIPTION

When **addbib** starts up, answering **y** to the initial **Instructions?** prompt yields directions; typing **n** or RETURN skips them. **addbib** then prompts for various bibliographic fields, reads responses from the terminal, and sends output records to *database*. A null response (just RETURN) means to leave out that field. A '–' (minus sign) means to go back to the previous field. A trailing backslash allows a field to be continued on the next line. The repeating **Continue?** prompt allows the user either to resume by typing **y** or RETURN, to quit the current session by typing **n** or **q**, or to edit *database* with any system editor (see **vi**(1), **ex**(1), **ed**(1)).

OPTIONS

–a Suppress prompting for an abstract; asking for an abstract is the default. Abstracts are ended with a CTRL–D.

–p *promptfile* Use a new prompting skeleton, defined in *promptfile*. This file should contain prompt strings, a TAB, and the key-letters to be written to the *database*.

USAGE

Bibliography Key Letters

The most common key-letters and their meanings are given below. **addbib** insulates you from these key-letters, since it gives you prompts in English, but if you edit the bibliography file later on, you will need to know this information.

%A	Author's name
%B	Book containing article referenced
%C	City (place of publication)
%D	Date of publication
%E	Editor of book containing article referenced
%F	Footnote number or label (supplied by **refer**)
%G	Government order number
%H	Header commentary, printed before reference
%I	Issuer (publisher)
%J	Journal containing article
%K	Keywords to use in locating reference
%L	Label field used by **–k** option of **refer**
%M	Bell Labs Memorandum (undefined)
%N	Number within volume
%O	Other commentary, printed at end of reference

%P	Page number(s)
%Q	Corporate or Foreign Author (unreversed)
%R	Report, paper, or thesis (unpublished)
%S	Series title
%T	Title of article or book
%V	Volume number
%X	Abstract — used by **roffbib**, not by **refer**
%Y,Z	Ignored by **refer**

EXAMPLES

Except for A, each field should be given just once. Only relevant fields should be supplied.

%A	Mark Twain
%T	Life on the Mississippi
%I	Penguin Books
%C	New York
%D	1978

SEE ALSO

ed(1), **ex**(1), **indxbib**(1), **lookbib**(1), **refer**(1), **roffbib**(1), **sortbib**(1), **vi**(1)

NAME

alias, unalias – shell built-in functions to create your own pseudonym or shorthand for a command or series of commands

SYNOPSIS

csh

alias [*name* [*def*]]

unalias *pattern*

ksh

†† **alias** [**–tx**] [*name*[=*value*]] . . .

unalias *name* .

DESCRIPTION

csh

alias assigns *def* to the alias *name*. *def* is a list of words that may contain escaped history-substitution metasyntax. *name* is not allowed to be **alias** or **unalias**. If *def* is omitted, the alias *name* is displayed along with its current definition. If both *name* and *def* are omitted, all aliases are displayed.

unalias discards aliases that match (filename substitution) *pattern*. All aliases may be removed by '**unalias** *****'.

ksh

alias with no arguments prints the list of aliases in the form *name=value* on standard output. An *alias* is defined for each name whose *value* is given. A trailing space in *value* causes the next word to be checked for alias substitution. The –t flag is used to set and list tracked aliases. The value of a tracked alias is the full pathname corresponding to the given *name*. The value becomes undefined when the value of **PATH** is reset but the aliases remained tracked. Without the –t flag, for each *name* in the argument list for which no *value* is given, the name and value of the alias is printed. The **–x** flag is used to set or print *exported aliases*. An *exported alias* is defined for scripts invoked by name. The exit status is non-zero if a *name* is given, but no value, and no alias has been defined for the *name*.

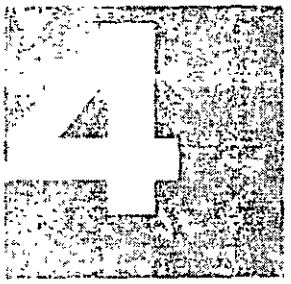
The *aliases* given by the list of *names* may be removed from the *alias* list with **unalias**.

On this man page, **ksh**(1) commands that are preceded by one or two † (daggers) are treated specially in the following ways:

1. Variable assignment lists preceding the command remain in effect when the command completes.
2. I/O redirections are processed after variable assignments.
3. Errors cause a script that contains them to abort.
4. Words, following a command preceded by †† that are in the format of a variable assignment, are expanded with the same rules as a variable assignment. This means that tilde substitution is performed after the = sign and word splitting and file name generation are not performed.

SEE ALSO

csh(1), **ksh**(1), **shell_builtins**(1),



INTRODUCCIÓN A UNIX

ADMINISTRACIÓN

PART I Managing User Accounts and Groups

This part provides instructions for managing users and groups. This part contains these chapters.

Chapter 1	Provides overview information about setting up user accounts and groups in a network environment.
Chapter 2	Provides step-by-step instructions for setting up user accounts and groups with Admintool.

CHAPTER 1

Managing User Accounts and Groups (Overview)

This chapter provides guidelines and planning information for managing user accounts and groups, and it provides overview information about setting up user accounts and groups in a network environment. This chapter also includes information about the files used to store user account and group information and about customizing the user's work environment.

This is a list of the overview information in this chapter.

- "What Are User Accounts and Groups?" on page 3
- "Guidelines for Managing User Accounts" on page 4
- "Guidelines for Managing Groups" on page 10
- "Tools for Managing User Accounts and Groups" on page 11
- "Where User Account and Group Information Is Stored" on page 16
- "Customizing a User's Work Environment" on page 20

For instructions about how to manage users accounts and groups, see Chapter 2.

What Are User Accounts and Groups?

One of the basic system administration tasks is to set up a user account for each user at a site. A typical user account includes the information a user needs to log in and use a system (without having the system's root password). User account information consists of four main components:

- User name – A name that a user uses to log in to a system (also known as a login name).

- Password – A secret combination of characters that a user must enter with a user name to gain access to a system.
- User's home directory – A directory that is usually the user's current directory at login. It typically contains most of the user's files.
- User initialization files – Shell scripts that control how the user's working environment is set up when a user logs in to a system.

Also, when you set up a user account, you can add the user to predefined groups of users. A typical use of groups is to set up file and directory access only to users who are part of a group (using the group permissions on a file or directory).

For example, you might have a directory containing top secret files that only a few users should be able to access. You could set up a group called `topsecret` that included the users working on the top secret project, and you could set up the top secret files with read permission for the `topsecret` group. That way, only the users in the `topsecret` group would be able to read the files.

Guidelines for Managing User Accounts

The following sections describe some guidelines and planning information for creating user accounts.

Name Services

If you are managing user accounts for a large site, you may want to consider using a name service such as NIS or NIS+. A name service enables you to store user account information in a centralized manner instead of storing user account information in every system's `/etc` files. When using a name service for user accounts, users can move from system to system using the same user account without having site-wide user account information duplicated in every system's `/etc` files. Using a name service also promotes centralized and consistent user account information.

User (Login) Names

User names, also called login names, let users access their own systems and remote systems that have the appropriate access privileges. You must choose a user name for each user account you create. User names must:

- Be unique within your organization, which may span multiple domains
- Contain from two to eight letters and numerals (the first character must be a letter and at least one character must be a lowercase letter)

- Not contain an underscore or space

It is helpful to establish a standard way of forming user names, and the names should be easy for users to remember. A simple scheme when selecting a user name is to use the first name initial and first seven letters of the user's last name. For example, Ziggy Ignatz becomes `zignatz`. If that scheme results in duplicate names, you can use the first initial, middle initial, and the first six characters of the user's last name. For example, Ziggy Top Ignatz becomes `ztignatz`. If that still results in duplicate names, you can use the first initial, middle initial, first five characters of the user's last name, and the number 1, or 2, or 3, and so on, until you have a unique name

---

Note - Each new user name must be distinct from any mail aliases known to the system or to an NIS or NIS+ domain. Otherwise, mail may be delivered to the alias rather than to the actual user.

User ID Numbers

Associated with each user name is a user identification (UID) number. The UID number identifies the user name to any system on which the user attempts to log in, and it is used by systems to identify the owners of files and directories. If you create user accounts for a single individual on a number of different systems, always use the same user name and user ID. In that way, the user can easily move files between systems without ownership problems.

UID numbers must be a whole number less than or equal to 2147483647, and they are required for both regular user accounts and special system accounts. Table 1–1 lists the UID numbers reserved for user accounts and system accounts.

TABLE 1–1 Reserved UID Numbers

User ID Numbers	Login Accounts	Reserved For ...
0 - 99	root, daemon, bin, sys, etc.	System accounts
100 - 2147483647	Regular users	General purpose accounts
60001	nobody	Unauthenticated users
60002	noaccess	Compatibility with Solaris 2.0 and compatible versions and SVR4 releases

Although UID numbers 0 through 99 are reserved, you can add a user with one of these numbers. However, do not use them for regular user accounts. By definition,

root always has UID 0, daemon has UID 1, and pseudo-user bin has UID 2. In addition, you should give uucp logins and pseudo user logins, like who, tty, and ttytype, low UIDs so they fall at the beginning of the passwd file.

As with user (login) names, you should adopt a scheme to assign unique UIDs. Some companies assign unique employee numbers, and administrators add 1000 to the employee number to create a unique UID number for each employee.

To minimize security risks, you should avoid reusing the UIDs from deleted accounts. If you must reuse a UID, "wipe the slate clean" so the new user is not affected by attributes set for a former user. For example, a former user may have been denied access to a printer—by being included in a printer deny list—but that attribute may not be appropriate for the new user. If need be, you can use duplicate UIDs in an NIS+ domain if the supply of unique UIDs is exhausted.

Using Large User IDs and Group IDs

Previous Solaris software releases used 32-bit data types to contain the user IDs (UIDs) and group IDs (GIDs), but UIDs and GIDs were constrained to a maximum useful value of 60000. Starting with the Solaris 2.5.1 release and compatible versions, the limit on UID and GID values has been raised to the maximum value of a signed integer, or 2147483647

UIDs and GIDs over 60000 do not have full functionality and are incompatible with many Solaris features, so avoid using UIDs or GIDs over 60000. See Table 1–2 for a complete list of interoperability issues with Solaris products and commands.

Table 1–2 describes interoperability issues with previous Solaris and Solaris product releases.

TABLE 1–2 Interoperability Issues for UIDs/GIDs Over 60000

Category	Product/Command	Issues/Cautions
NFS™ Interoperability	SunOS 4.0 NFS software and compatible versions	NFS server and client code truncates large UIDs and GIDs to 16 bits. This can create security problems if SunOS 4.0 and compatible machines are used in an environment where large UIDs and GIDs are being used. SunOS 4.0 and compatible systems require a patch.
Name Service Interoperability	NIS name service File-based name service	Users with UIDs above 60000 can log in or use the su command on systems running the Solaris 2.5 and compatible versions, but their UIDs and GIDs will be set to 60001 (nobody).

TABLE 1–2 Interoperability Issues for UIDs/GIDs Over 60000 *(continued)*

Category	Product/Command	Issues/Cautions
	NIS– name service	Users with UIDs above 60000 are denied access on systems running Solaris 2.5 and compatible versions and the NIS+ name service.
Printed UIDs/GIDs	OpenWindows File Manager	Large UIDs and GIDs will not display correctly if the OpenWindows™ File Manager is used with the extended file listing display option.

TABLE 1–3 Large UID/GID Limitation Summary

A UID or GID Of ...	Limitations
60003 or greater	■ Users in this category logging into systems running Solaris 2.5 and compatible releases and the NIS or files name service will get a UID and GID of `nobody`
65535 or greater	■ Solaris 2.5 and compatible releases systems running the NFS version 2 software will see UIDs in this category truncated to 16 bits, creating possible security problems. ■ Users in this category using the `cpio` command (using the default archive format) to copy files will see an error message for each file and the UIDs and GIDs will be set to `nobody` in the archive ■ SPARC systems: Users in this category running SunOS 4.0 and compatible applications will see `EOVERFLOW` returns from some system calls, and their UIDs and GIDs will be mapped to `nobody` ■ x86 systems: Users in this category on x86 systems running SVR3-compatible applications will probably see `EOVERFLOW` return codes from system calls ■ x86 systems: If users in this category attempt to create a file or directory on a mounted System V file system, the System V file system returns an `EOVERFLOW` error
100000 or greater	■ The `ps -l` command displays a maximum five-digit UID so the printed column won't be aligned when they include a UID or GID larger than 99999
262144 or greater	■ Users in this category using the `cpio` command (using `-H odc` format) or the `pax -x cpio` command to copy files will see an error message returned for each file, and the UIDs and GIDs will be set to `nobody` in the archive.

TABLE 1–3 Large UID/GID Limitation Summary *(continued)*

A UID or GID Of ...	Limitations
1000000 or greater	■ Users in this category using the `ar` command will have their UIDs and GIDs set to `nobody` in the archive.
2097152 or greater	■ Users in this category using the `tar` command, the `cpio -H ustar` command, or the `pax -x tar` command have their UIDs and GIDs set to `nobody`.

Passwords

Although user names are publicly known, passwords must be kept secret and known only to users. Each user account should be assigned a password, which is a combination of six to eight letters, numbers, or special characters. You can set a user's password when you create the user account and have the user change it when logging in to a system for the first time.

To make your computer systems more secure, ask users to change their passwords periodically. For a high level of security, you should require users to change their passwords every six weeks. Once every three months is adequate for lower levels of security. System administration logins (such as root and sys) should be changed monthly, or whenever a person who knows the root password leaves the company or is reassigned.

Many breaches of computer security involve guessing a legitimate user's password. You should make sure that users avoid using proper nouns, names, login names, and other passwords that a person might guess just by knowing something about the user.

Good choices for passwords include:

- Phrases (`beammeup`)
- Nonsense words made up of the first letters of every word in a phrase (`swotrb` for SomeWhere Over The RainBow)
- Words with numbers or symbols substituted for letters (`sn00py` for snoopy)

Do not use these choices for passwords:

- Your name, forwards, backwards, or jumbled
- Names of family members or pets
- Car license numbers
- Telephone numbers
- Social Security numbers

- Employee numbers
- Names related to a hobby or interest
- Seasonal themes, such as Santa in December
- Any word in the dictionary

Password Aging

If you are using NIS+ or the `/etc` files to store user account information, you can set up password aging on a user's password. Password aging enables you to force users to change their passwords periodically or to prevent a user from changing a password before a specified interval. If you want to prevent an intruder from gaining undetected access to the system by using an old and inactive account, you can also set a password expiration date when the account will be disabled.

Home Directories

The home directory is the portion of a file system allocated to a user for storing private files. The amount of space you allocate for a home directory depends on the kinds of files the user creates and the type of work done. As a general rule, you should allocate at least 15 Mbytes of disk space for each user's home directory.

A home directory can be located either on the user's local system or on a remote file server. In either case, by convention the home directory should be created as `/export/home/`*username*. For a large site, you should store home directories on a server. Use a separate file system for each `/export/home`*n* directory to facilitate backing up and restoring home directories (for example, `/export/home1`, `/export/home2`).

Regardless of where their home directory is located, users usually access their home directories through a mount point named `/home/`*username*. When AutoFS is used to mount home directories, you are not permitted to create any directories under the `/home` mount point on any system. The system recognizes the special status of `/home` when Autofs is active. For more information about automounting home directories, see the *NFS Administration Guide*.

To use the home directory anywhere on the network, you should always refer to it as `$HOME`, not as `/export/home/`*username*. The latter is machine-specific. In addition, any symbolic links created in a user's home directory should use relative paths (for example, `../../../x/y/x`), so the links will be valid no matter where the home directory is mounted.

The User's Work Environment

Besides having a home directory to create and store files, users need an environment that gives them access to the tools and resources they need to do their work. When a user logs in to a system, the user's work environment is determined by initialization files that are defined by the user's startup shell, such as the C, Korn, or Bourne shell.

A good strategy for managing the user's work environment is to provide customized user initialization files (`.login`, `.cshrc`, `.profile`) in the user's home directory. See "Customizing a User's Work Environment" on page 20 for detailed information about customizing user initialization files for users. After you create the customized user initialization files, you can add them to a user's home directory when you create a new user account.

A recommended one-time task is to set up separate directories, called skeleton directories, on a server (you can use the same server where the user's home directories are stored). The skeleton directories enable you to store customized user initialization files for different types of users.

Note - Do not use system initialization files (`/etc/profile`, `/etc/.login`) to manage a user's work environment, because they reside locally on systems and are not centrally administered. For example, if AutoFS is used to mount the user's home directory from any system on the network, then you would have to modify the system initialization files on each system to ensure a consistent environment when a user moves from system to system.

Guidelines for Managing Groups

A *group* is a collection of users who can share files and other system resources. For example, the set of users working on the same project could be formed into a group. A group is traditionally known as a UNIX group.

Each group must have a name, a group identification (GID) number, and a list of user names that belong to the group. A GID identifies the group internally to the system. There are two types of groups that a user can belong to:

- Primary group – Specifies a group that the operating system will assign to files created by the user. Each user must belong to a primary group.
- Secondary groups – Specifies one or more groups to which a user also belongs. Users can belong to up to 16 secondary groups.

Sometimes a user's secondary group is not important. For example, ownership of files reflect the primary group, not any secondary groups. Other applications, however, may rely on a user's secondary memberships. For example, a user has to be

a member of the sysadmin group (group 14) to use the Solstice AdminSuite software, but it doesn't matter if group 14 is his or her current primary group.

The `groups` command shows the groups a user belongs to. A user can have only one primary group at a time. However, the user can temporarily change the primary group (with the `newgrp` command) to any other group in which he or she is a member.

When adding a user account, you must assign a primary group for a user or accept the default `staff` (group 10). The primary group should already exist (if it doesn't exist, specify the group by a GID number). User names are not added to primary groups. If they were, the list might become too long. Before you can assign users to a new secondary group, you must create the group and assign it a GID number.

Groups can be local to a system or can be managed through a name service. To simplify group administration, you should use a name service like NIS+, which enables you to centrally manage group memberships.

Tools for Managing User Accounts and Groups

Table 1–4 lists the recommended tools for managing users and groups.

TABLE 1–4 Recommended Tools for Managing Users and Groups

If You Are Managing Users and Groups ...	The Recommended Tool Is ...	And You Will Need ...	To Start This Tool See ...
On remote and/or local systems in a networked, name service (NIS, NIS+) environment	Solstice™ AdminSuite™'s User and Group Manager (graphical user interface)	Graphics monitor running an X window such as CDE or OpenWindows	*Solstice AdminSuite 2.3 Administration Guide*
On a local system	Admintool (graphical user interface)	Graphics monitor running an X window system such as CDE or OpenWindows	Chapter 2

The Solaris commands `useradd` and `groupadd` also let you set up users and groups on a local system; however, the commands do not change name service maps or tables. Table 1–5 describes the Solaris command used to manage user accounts and groups if you are not using Solstice AdminSuite or Admintool.

TABLE 1–5 Managing User Accounts and Groups by Using Solaris Commands

Task	If You Use This Name Service ...	Then Use These Commands
Add a User Account	NIS+	`nistbladm`
		`nisclient`
	NIS	`useradd`
		`make`
	None	`useradd`
Modify a User Account	NIS+	`nistbladm`
	NIS	`usermod`
		`make`
	None	`usermod`
Delete a User Account	NIS+	`nistbladm`
		`nisclient`
	NIS	`userdel`
		`make`
	None	`userdel`
Set Up User Account Defaults	NIS+	not available
	NIS	`useradd -D`
		`make`
	None	`useradd -D`
Disable a User Account	NIS+	`nistbladm`

TABLE 1–5 Managing User Accounts and Groups by Using Solaris Commands *(continued)*

Task	If You Use This Name Service ...	Then Use These Commands
	NIS	`passwd -r nis -l`
		`make`
	None	`passwd -r files -l`
Change a User's Password	NIS+	`passwd -r nisplus`
	NIS	`passwd -r nis`
	None	`passwd -r files`
Sort User Accounts	NIS+	`niscat`
		`sort`
	NIS	`ypcat`
		`sort`
	None	`awk`
		`sort`
Find a User Account	NIS+	`nismatch`
	NIS	`ypmatch`
	None	`grep`
Add a Group	NIS+	`nistbladm`
	NIS	`groupadd`
		`make`
	None	`groupadd`
Modify Users in a Group	NIS+	`nistbladm`
	NIS	`groupmod`
		`make`

TABLE 1–5 Managing User Accounts and Groups by Using Solaris Commands *(continued)*

Task	If You Use This Name Service ...	Then Use These Commands
Delete a Group	None	`groupmod`
	NIS+	`nistbladm`
	NIS	`groupdel` `make`
	None	`groupdel`

What You Can Do With Admintool

Admintool is a graphical user interface that enables you to set up user accounts on a local system.

Modify User Accounts

Unless you define a user name or UID number that conflicts with an existing one, you should never need to modify a user account's login name or UID number. Use the following steps if two user accounts have duplicate user names or UID numbers:

- If two user accounts have duplicate UID numbers, use Admintool to remove one account and re-add it with a different UID number. You cannot use Admintool to modify a UID number of an existing user account.
- If two user account have duplicate user names, use Admintool to modify one of the accounts and change the user name.

  If you do use Admintool to change a user name, the home directory's ownership is changed (if a home directory exists for the user).

One part of a user account that you can change is a user's group memberships Admintool Modify option lets you add or delete a user's secondary groups. Alternatively, you can use the Groups window to directly modify a group's member list.

You can also modify the following parts of a user account.

- Comment
- Login shell
- Passwords
- Home directory
- AutoHome setup
- Credential table setup
- Mail server

Delete User Accounts

When you delete a user account with Admintool, the software deletes the entries in the `passwd` and `group` files. In addition, you can delete the files in the user's home directory and delete the contents of the user's mailbox.

Only the single mail alias that directs mail to the user's mail box is deleted; the user name is not deleted from any other mail aliases. If you want to delete entries from mail aliases other than the one set up to direct mail to your mailbox, you must delete them by hand.

Add Customized User Initialization Files

Although you can't create customized user initialization files with Admintool, you can populate a user's home directory with user initialization files located in a specified "skeleton" directory.

You can customize the user initialization templates in the `/etc/skel` directory and then copy them to users' home directories.

Administer Passwords

You can use Admintool for password administration, which includes specifying a normal password for a user account, enabling users to create their own passwords during their first login, disabling or locking a user account, or specifying expiration dates and password aging information.

Note - Password aging is not supported by the NIS name service.

Disable User Accounts

Occasionally, you may need to temporarily or permanently disable a login account. Disabling or locking a user account means that an invalid password, `*LK*`, is assigned to the user account, preventing future logins.

The easiest way to disable a user account is to use Admintool to lock the password for an account. You can also enter an expiration date in the Expiration Date field to set how long the user account is disabled.

Other ways to disable a user account is to set up password aging or by just changing the user's password.

Where User Account and Group Information Is Stored

Depending on your site policy, you can store user account and group information in a name service or a local system's `/etc` files. In the NIS+ name service, information is stored in tables, and in the NIS name service, information is stored in maps.

Note - To avoid confusion, the location of the user account and group information will be generically referred to as a *file* rather than a *file*, *table*, or *map*.

Most of the user account information is stored in the `passwd` file. However, password encryption and password aging is stored in the `passwd` file when using NIS or NIS+ and in the `/etc/shadow` file when using `/etc` files. Password aging is not available when using NIS.

Group information is stored in the `group` file.

Fields in the `passwd` File

The fields in the `passwd` file are separated by colons and contain the following information:

```
username password: uid:gid:comment home-directory:login-shell
```

For example:

```
kryten:x:101:100:Kryten Series 4000:/export/home/kryten:/bin/csh
```

Table 1-6 describes the `passwd` file fields.

TABLE 1–6 Fields in the passwd File

Field Name	Description
username	Contains the user or login name. User names should be unique and consist of 1-8 letters (A-Z, a-z) and numerals (0-9) The first character must be a letter, and at least one character must be a lowercase letter. User names cannot contain underscores or spaces.
password	Contains an x, a placeholder for the encrypted password. The encrypted password is stored in the shadow file.
uid	Contains a user identification (UID) number that identifies the user to the system. UID numbers for regular users should range from 100 to 60000. All UID numbers should be unique
gid	Contains a group identification (GID) number that identifies the user's primary group. Each GID number must be a whole number between 0 and 60002 (60001 and 60002 are assigned to nobody and noaccess, respectively)
comment	Usually contains the full name of the user (This field is informational only.) It is sometimes called the GECOS field because it was originally used to hold the login information needed to submit batch jobs to a mainframe running GECOS (General Electric Computer Operating System) from UNIX systems at Bell Labs.
home-directory	Contains user's home directory path name.
login-shell	Contains the user's default login shell, which can be /bin/sh, /bin/csh or /bin/ksh Table 1-11 contains a description of shell features.

Fields in the Shadow File

The fields in the shadow file are separated by colons and contain the following information

```
username:password lastchg:min:max:warn:inactive:expire
```

For example:

```
rimmer:86Kg/MNT/dGu.:8882:0::5:20:8978
```

Table 1–7 describes the shadow file fields.

TABLE 1–7 Fields in the shadow File

Field Name	Description
username	Contains the user or login name.
password	May contain the following entries: a 13-character encrypted user password; the string `*LK*`, which indicates an inaccessible account; or the string `NP`, which indicates no password for the account.
lastchg	Indicates the number of days between January 1, 1970, and the last password modification date
min	Contains the minimum number of days required between password changes
max	Contains the maximum number of days the password is valid before the user is prompted to specify a new password.
inactive	Contains the number of days a user account can be inactive before being locked
expire	Contains the absolute date when the user account expires. Past this date, the user cannot log in to the system.

Fields in the Group File

The fields in the `group` file are separated by colons and contain the following information:

```
group-name:group-password:gid user-list
```

For example:

```
bin::2:root,bin,daemon
```

Table 1–8 describes the `group` file fields.

TABLE 1–8 Fields in the group File

Field Name	Description
group-name	Contains the name assigned to the group. For example, members of the chemistry department in a university may be called chem. Group names can have a maximum of nine characters.
group-password	Usually contains an asterisk or is empty. The *group-password* field is a relic of earlier versions of UNIX. If a group has a password, the newgrp command prompts users to enter it. However, there is no utility to set the password
gid	Contains the group's GID number. It must be unique on the local system, and should be unique across the entire organization. Each GID number must be a whole number between 0 and 60002. Numbers under 100 are reserved for system default group accounts. User defined groups can range from 100 to 60000 (60001 and 60002 are reserved and assigned to nobody and noaccess, respectively.)
user-list	Contains a list of comma-separated list of user names, representing the user's secondary group memberships Each user can belong to a maximum of 16 secondary groups.

UNIX User Groups

By default, all Solaris 7 systems have these groups:

```
root::0:root
other::1:
bin::2:root,bin,daemon
sys::3:root,bin,sys,adm
adm::4:root,adm,daemon
uucp::5:root,uucp
mail::6,root
tty::7:root,tty,adm
lp::8:root,lp,adm
nuucp::9:root,nuucp
staff::10:
daemon::12:root,daemon
sysadmin: 14:
nobody :60001,
noaccess,:60002,
nogroup::65534:
```

Customizing a User's Work Environment

Part of setting up a user's home directory is providing user initialization files for the user's login shell. A *user initialization file* is a shell script that sets up a work environment for a user after the user logs in to a system. Basically, you can perform any task in a user initialization file that you can do in a shell script, but its primary job is to define the characteristics of a user's work environment, such as a user's search path, environment variables, and windowing environment. Each login shell has its own user initialization file (or files), which are listed in Table 1–9.

TABLE 1–9 User Initialization Files for Bourne, C, and Korn Shells

Shell	User Initialization File	Purpose
Bourne	`$HOME/.profile`	Defines user's environment at login
C	`$HOME/.cshrc`	Defines user's environment for all C shells, invoked after login shell
	`$HOME/.login`	Defines user's environment at login
Korn	`$HOME/.profile`	Defines user's environment at login
	`$HOME/$ENV`	Defines user's environment at login in the file, specified by the Korn shell's `ENV` environment variable

The Solaris system software provides default user initialization files for each shell in the `/etc/skel` directory on each system, as shown in Table 1–10.

TABLE 1–10 Default User Initialization Files

Shell	Default File
C	`/etc/skel/local.login`
	`/etc/skel/local.cshrc`
Bourne or Korn	`/etc/skel/local.profile`

TABLE 1–10 Default User Initialization Files *(continued)*

You can use these as a starting point and modify them to create a standard set of files that will provide the work environment common to all users, or you can modify them to provide the working environment for different types of users. See "How to Customize User Initialization Files" on page 36 for step-by-step instructions on how to create sets of user initialization files for different types of users.

Use Site Initialization Files

When customizing a user initialization file, it is important that the user initialization files can be customized by both the administrator and the user. This important feature can be accomplished with centrally located and globally distributed user initialization files, called site initialization files. Site initialization files enable you to continually introduce new functionality to the user's work environment, while enabling the user to also customize the user initialization file.

When you reference a site initialization file in a user initialization file, all updates to the site initialization file are automatically reflected when the user logs in to the system or when a user starts a new shell. Site initialization files are designed for you to distribute site-wide changes to users' work environments that you did not anticipate when you added the users.

Any customization that can be done in a user initialization file can be done in a site initialization file. These files typically reside on a server (or set of servers), and appear as the first statement in a user initialization file. Also, each site initialization file must be the same type of shell script as the user initialization file that references it.

To reference a site initialization file in a C-shell user initialization file, place a line similar to the following at the beginning of the user initialization file:

```
source /net/machine-name/export/site-files/site-init-file
```

To reference a site initialization file in a Bourne- or Korn-shell user initialization file, place a line similar to the following at the beginning of the user initialization file:

```
. /net/machine-name/export/site-files/site-init-file
```

Avoid Local System References

You should not add specific references to the local system in the user's initialization file. You want the instructions in a user initialization file to be valid regardless of the system to which the user logs in. For example:

- To make a user's home directory available anywhere on the network, always refer to the home directory with the variable `$HOME`. For example, use `$HOME/bin`; do not use `/export/home/`*`username`*`/bin`. `$HOME` works when the user logs in to another system, when home directories are automounted.
- To access files on a local disk, use global path names, like `/net/`*`machine-name`*`/`*`directory-name`*. Any directory referenced by `/net/`*`machine-name`* can be mounted automatically on any system on which the user logs in, assuming the system is running AutoFS.

Shell Features

Table 1–11 lists basic shell features that each shell provides, which can help you determine what you can and can't do when creating user initialization files for each shell.

TABLE 1–11 Basic Features of Bourne, C, and Korn Shells

Feature	Bourne	C	Korn
Known as the standard shell in UNIX	Yes	No	No
Compatible syntax with Bourne shell	-	No	Yes
Job control	Yes	Yes	Yes
History list	No	Yes	Yes
Command-line editing	No	Yes	Yes
Aliases	No	Yes	Yes
Single-character abbreviation for login directory	No	Yes	Yes
Protection from overwriting (`noclobber`)	No	Yes	Yes
Setting to ignore Control-d (`ignoreeof`)	No	Yes	Yes
Enhanced cd	No	Yes	Yes

TABLE 1–11 Basic Features of Bourne, C, and Korn Shells *(continued)*

Feature	Bourne	C	Korn
Initialization file separate from `.profile`	No	Yes	Yes
Logout file	No	Yes	No

Shell Environment

A shell maintains an environment that includes a set of variables defined by the `login` program, the system initialization file, and the user initialization files. In addition, some variables are defined by default. A shell can have two types of variables:

- Environment variables – Variables that are exported to all processes spawned by the shell. Their settings can be seen with the `env` command. A subset of environment variables, like `PATH`, affects the behavior of the shell itself.
- Shell (local) variables – Variables that affect only the current shell. In the C shell, a set of these shell variables have a special relationship to a corresponding set of environment variables. These shell variables are `user`, `term`, `home`, and `path`. The value of the environment variable counterpart is initially used to set the shell variable.

In the C shell, you use the lowercase names with the `set` command to set shell variables and use uppercase names with the `setenv` command to set environment variables. If you set a shell variable, the shell sets the corresponding environment variable and vice versa. For example, if you update the `path` shell variable with a new path, the shell also updates the `PATH` environment variable with the new path.

In the Bourne and Korn shells, you use the uppercase names with the `setenv` command to set both shell and environment variables. You also have to use the `export` command to finish setting environment variables. For all shells, you generally refer to shell and environment variables by their uppercase names.

In a user initialization file, you can customize a user's shell environment by changing the values of the predefined variables or by specifying additional variables. Table 1–12 shows how to set environment variables in a user initialization file

TABLE 1–12 Setting Environment Variables in a User Initialization File

If You Want to Set a User's Environment Variables for The ...	Then Add the Following Line to the User Initialization File ...
C shell	`setenv` *VARIABLE value* Example: `setenv MAIL /var/mail/ripley`
Bourne or Korn shell	*VARIABLE=value*`; export` *VARIABLE* Example: `MAIL=/var/mail/ripley;export MAIL`

Table 1–13 describes environment and shell variables you may want to customize in a user initialization file. For more information about variables used by the different shells, see **sh**(1), **ksh**(1), or **csh**(1).

TABLE 1–13 Shell and Environment Variable Descriptions

Variable	Description
`ARCH`	Sets the user's system architecture (for example, sun4, `i386`). This variable can be set with `ARCH = 'uname -p'` (in Bourne or Korn shells) or `setenv ARCH 'uname -p'` (in C shell). There is no built-in behavior of the shell that depends on this variable. It's just a useful variable for branching within shell scripts.
`CALENDAR`	Sets the path to the Calendar executables
`CDPATH` (or `cdpath` in the C shell)	Sets a variable used by the `cd` command. If the target directory of the `cd` command is specified as a relative path name, the `cd` command will first look for the target directory in the current directory (".") If the target is not found, the path names listed in the `CDPATH` variable are searched consecutively until the target directory is found and the directory change is completed. If the target directory is not found, the current working directory is left unmodified. For example, the `CDPATH` variable is set to `/home/jean`, and two directories exist under `/home/jean` `bin` and `rje`. If you are in the `/home/jean/bin` directory and type `cd rje`, you change directories to `/home/jean/rje`, even though you do not specify a full path.

TABLE 1–13 Shell and Environment Variable Descriptions *(continued)*

Variable	Description
DESKSET	Sets the path to the DeskSet™ executables.
history	Sets history for the C shell
HOME (or home in the C shell)	Sets the path to the user's home directory.
LANG	Sets the locale.
LOGNAME	Defines the name of the user currently logged in. The default value of LOGNAME is automatically set by the login program to the user name specified in the passwd file. You should only need to refer to (not reset) this variable.
LPDEST	Sets the user's default printer.
MAIL	Sets the path to the user's mailbox.
MANPATH	Sets the hierarchies of man pages available.
MANSECTS	Sets the hierarchies of man pages available.
OPENWINHOME	Sets the path to the OpenWindows subsystem
PATH (or path in the C shell)	Lists, in order, the directories that the shell searches to find the program to run when the user types a command. If the directory is not in the search path, users must type the complete path name of a command. The default PATH is automatically defined and set as specified in .profile (Bourne or Korn shell) or .cshrc (C shell) as part of the login process. The order of the search path is important. When identical commands exist in different locations, the first command found with that name is used. For example, suppose that PATH is defined (in Bourne and Korn shell syntax) as PATH=/bin:/usr/bin:/usr/sbin:$HOME/bin and a file named sample resides in both /usr/bin and /home/jean/bin. If the user types the command sample without specifying its full path name, the version found in /usr/bin is used
prompt	Defines the shell prompt for the C shell
PS1	Defines the shell prompt for the Bourne or Korn shell.

TABLE 1–13 Shell and Environment Variable Descriptions *(continued)*

Variable	Description
SHELL (or shell in the C shell)	Sets the default shell used by `make`, `vi`, and other tools.
TERMINFO	Specifies the path name for an unsupported terminal that has been added to the `terminfo` file. Use the TERMINFO variable in `/etc/profile` or `/etc/.login`. When the TERMINFO environment variable is set, the system first checks the TERMINFO path defined by the user. If it does not find a definition for a terminal in the TERMINFO directory defined by the user, it searches the default directory, `/usr/share/lib/terminfo`, for a definition. If it does not find a definition in either location, the terminal is identified as "dumb "
TERM (or term in the C shell)	Defines the terminal. This variable should be reset in `/etc/profile` or `/etc/.login`. When the user invokes an editor, the system looks for a file with the same name as the definition of this environment variable. The system searches the directory referenced by TERMINFO to determine the terminal characteristics.
TZ	Sets the time zone, which is used to display dates, for example, in the `ls -l` command. If TZ is not set in the user's environment, the system setting is used; otherwise, Greenwich Mean Time is used.

The PATH Variable

When the user executes a command by using the full path, the shell uses that path to find the command. However, when users specify only a command name, the shell searches the directories for the command in the order specified by the PATH variable. If the command is found in one of the directories, the shell executes it.

A default path is set by the system, but most users modify it to add other command directories. Many user problems related to setting up the environment and accessing the right version of a command or a tool can be traced to incorrectly defined paths.

Guidelines

Here are some guidelines for setting up efficient PATH variables:

- If security is not a concern, put the current working directory (.) first in the path. However, including the current working directory in the path poses a security risk that you may want to avoid, especially for superuser.

- Keep the search path as short as possible. The shell searches each directory in the path. If a command is not found, long searches can slow down system performance.
- The search path is read from left to right, so you should put directories for commonly used commands at the beginning of the path.
- Make sure directories are not duplicated in the path.
- Avoid searching large directories, if possible Put large directories at the end of the path.
- Put local directories before NFS™ mounted directories to lessen the chance of "hanging" when the NFS server does not respond and to reduce unnecessary network traffic.

Examples—Setting a User's Default Path

The following examples show how to set a user's default path to include the home directory and other NFS mounted directories (the current working directory is specified first in the path). In a C-shell user initialization file, you would add the following:

```
set path=(. /usr/bin $HOME/bin /net/glrr/files1/bin)
```

In a Bourne- or Korn-shell user initialization file, you would add the following:

```
PATH=.:/usr/bin /$HOME/bin:/net/glrr/files1/bin
export PATH
```

The Locale Variables

The LANG and LC environment variables specify the locale-specific conversions and conventions for the shell, like timezones, collation orders, and formats of dates, time, currency, and numbers. In addition, you can use the `stty` command in a user initialization file to set whether the system will support multibyte characters.

LANG sets all possible conversions and conventions for the given locale. If you have special needs, you can set various aspects of localization separately through these LC variables: LC_COLLATE, LC_CTYPE, LC_MESSAGES, LC_NUMERIC, LC_MONETARY, and LC_TIME.

Table 1-14 describes the values for the LANG and LC environment variables.

TABLE 1–14 Values for LANG and LC Variables

Value	Locale
de	German
fr	French
iso_8859_1	English and European
it	Italian
japanese	Japanese
korean	Korean
sv	Swedish
tchinese	Taiwanese

Examples—Setting the Locale Using the LANG Variables

The following examples show how to set the locale using the LANG environment variables. In a C-shell user initialization file, you would add the following:

```
setenv LANG DE
```

In a Bourne- or Korn-shell user initialization file, you would add the following:

```
LANG=DE, export LANG
```

Default File Permissions (umask)

When you create a file or directory, the default file permissions assigned to the file or directory are controlled by the *user mask*. The user mask should be set by the umask command in a user initialization file. You can display the current value of the user mask by typing umask and pressing Return.

The user mask can be set with a three-digit octal value. The first digit sets permissions for the user; the second sets permissions for group; the third sets permissions for other (also referred to as "world"). Note that if the first digit is zero, it is not displayed. For example, if umask is set to 022, 22 is displayed.

To determine the umask value you want to set, subtract the value of the permissions you want from 666 (for a file) or 777 (for a directory). The remainder is the value to use with the umask command. For example, suppose you want to change the default mode for files to 644 (rw-r--r--). The difference between 666 and 644 is 022, which is the value you would use as an argument to the umask command

You can also determine the umask value you want to set by using Table 1–15, which shows the file and directory permissions that are created for each of the octal values of umask.

TABLE 1–15 Permissions for umask Values

umask Octal Value	File Permissions	Directory Permissions
0	rw-	rwx
1	rw-	rw-
2	r--	r-x
3	r--	r--
4	-w-	-wx
5	-w-	-w-
6	--x	--x
7	--- (none)	--- (none)

The following line in a user initialization file sets the default file permissions to rw-rw-rw-

```
umask 000
```

Examples of User and Site Initialization Files

The following sections provide examples of user and site initialization files that you can use to start customizing your own initialization files. Many of the examples use system names and paths that you will have to change for your particular site.

Example—.profile File

CODE EXAMPLE 1–1 Example .profile File

```
1 PATH=$PATH:$HOME/bin:/usr/local/bin:/usr/ccs/bin:.
2 MAIL=/var/mail/$LOGNAME
3 NNTPSERVER=server1
4 MANPATH=/usr/share/man:/usr/local/man
5 PRINTER=printer1
6 umask 022
7 export PATH MAIL NNTPSERVER MANPATH PRINTER
```

Example—.cshrc File

CODE EXAMPLE 1–2 Example .cshrc File

```
8 set path=($PATH $HOME/bin /usr/local/bin /usr/ccs/bin)
9 setenv MAIL /var/mail/$LOGNAME
10 setenv NNTPSERVER server1
11 setenv PRINTER printer1
12 alias h history
13 umask 022
14 source /net/server2/site-init-files/site.login
```

Example—Site Initialization File

The following shows an example site initialization file in which a user can choose a particular version of an application.

---

1 Defines the user's shell search path
2 Defines the path to the user's mail file
3 Defines the environment variable to the user's Usenet news server
4 Defines the user's search path for man pages
5 Defines the user's default printer
6. Sets the user's default file creation permissions
7. Sets the listed environment variables
8 Sets the user's shell search path
9 Sets the path to the user's mail file
10 Sets the user's Usenet news server.
11. Sets the user's default printer.
12 Creates an alias for the history command (the user will need to type only h to run the history command)

TABLE 1–16 Example Site Initialization File

```
# @(#)site.login
main:
echo "Application Environment Selection"
echo ""
echo "1. Application, Version 1"
echo "2. Application, Version 2"
echo ""
echo -n "Type 1 or 2 and press Return to set your
application environment: "

set choice = $<

if ( $choice !~ [1-2] ) then
goto main
endif

switch ($choice)

case "1":
setenv APPHOME /opt/app-v.1
breaksw

case "2":
setenv APPHOME /opt/app-v.2
endsw
```

This site initialization file could be referenced in a user's `.cshrc` file (C shell users only) with the following line:

```
source /net/server2/site-init-files/site.login
```

In this line, the site initialization file is named `site.login` and is located on a server named `server2`. This line also assumes that the automounter is running on the user's system.

13 Sets the user's default file creation permissions
14 Runs the site initialization file shown in "Example—Site Initialization File" on page 30

CHAPTER 2

Setting Up and Maintaining User Accounts and Groups (Tasks)

This chapter describes the procedures for setting up and maintaining user accounts and groups.

This is a list of the step-by-step instructions in this chapter.

- "How to Customize User Initialization Files" on page 36
- "How to Start Admintool" on page 37
- "How to Add a Group" on page 38
- "How to Add a New User Account" on page 39
- "How to Share a User's Home Directory" on page 41
- "How to Mount a User's Home Directory" on page 43
- "How to Modify a Group" on page 45
- "How to Delete a Group" on page 46
- "How to Modify a User Account" on page 46
- "How to Disable a User Account " on page 48
- "How to Change a User's Password " on page 49
- "How to Change Password Aging for a User Account " on page 50
- "How to Delete a User Account " on page 51
- "How to Restart Solaris User Registration" on page 54
- "How To Disable User Registration" on page 55

For overview information about Managing User Accounts and Groups, see Chapter 1.

Becoming Superuser (root)

Most administrative tasks such as adding users require that you log in as root (UID=0) first. The root account is also known as the *superuser* account because it's used to make system changes and can override user file protection in emergency situations.

The superuser account should only be used to perform administrative tasks to prevent indiscriminate changes to the system.

You can either log into the system as superuser or use the **su**(1M) command to change to the superuser account.

▼ How to Become Superuser (root)

Become superuser by one of the following methods. Both methods require that you know the root password.

- Change to the superuser account by using the su command.

```
% su
Password: root_password
#
```

The pound sign (#) is the Bourne shell prompt for the superuser account.

- Log in as superuser on the system console.

```
hostname console: root
Password: root_password
#
```

This method is not enabled by default. You must modify the `/etc/default/login` file to log in as superuser on the system console. See "Securing Systems (Tasks)" in *System Administration Guide, Volume II* for information on modifying this file.

Setting Up User Accounts Task Map

TABLE 2–1 Task Map: Setting Up User Accounts

Task	Description	For Instructions, Go To
1. Customize User Initialization Files	*Optional.* Set up user initialization files (`.cshrc`, `.profile`, `.login`), so you can provide new users with consistent environments	"How to Customize User Initialization Files" on page 36
2. Add a Group	*Optional.* To help administer users, add groups by using the Groups main window	"How to Add a Group" on page 38
3. Add a User Account	Add a user account by using Admintool's Users main window.	"How to Add a New User Account" on page 39
4. Share the User's Home Directory	Share the user's home directory, so the directory can be remotely mounted from the user's system.	"How to Share a User's Home Directory" on page 41
5. Mount the User's Home Directory	Manually mount the user's home directory on the user's system by using the `mount` command.	"How to Mount a User's Home Directory" on page 43

User Information Data Sheet

You may find it useful to create a form like the one below to gather information about users before adding their accounts

Item	Description
User Name:	
UID:	
Primary Group:	
Secondary Groups	
Comment:	
Default Shell	
Password Status and Aging	

Home Directory Server Name:	
Home Directory Path Name:	
Mounting Method:	
Permissions on Home Directory:	
Mail Server:	
Department Name:	
Department Administrator:	
Manager.	
Employee Name:	
Employee Title:	
Employee Status:	
Employee Number:	
Start Date:	
Add to These Mail Aliases:	
Desktop System Name:	

▼ How to Customize User Initialization Files

1. **Become superuser on the system where the users' home directories are created and shared.**

2. **Create a skeleton directory for each type of user.**

```
# mkdir /shared-directory/skel/user-type
```

shared-directory	The name of a directory that is available to other systems on the network.
user-type	The name of a directory to store initialization files for a type of user

3. **Copy the default user initialization files into the directories you created for different types of users.**

```
# cp /etc/skel/local.cshrc /shared-directory/skel/user-type/.cshrc
# cp /etc/skel/local.login /shared-directory/skel/user-type/.login
# cp /etc/skel/local.profile /shared-directory/skel/user-type
/.profile
```

Note - You can use the `ls -a` command to list . (dot) files.

4. **Edit the user initialization files for each user type and customize them based on your site's needs.**

   See "Customizing a User's Work Environment" on page 20 for a detailed description on the ways to customize the user initialization files.

5. **Set the permissions for the user initialization files.**

```
# chmod 744 /shared-directory/skel/user-type/.*
```

Example—Customizing User Initialization Files

The following example customizes the C-shell user initialization file in the `/export/skel/enduser` directory designated for a particular type of user.

```
# mkdir /export/skel/enduser
# cp /etc/skel/local.cshrc /export/skel/enduser/.cshrc

(Edit .cshrc file-see "Example—.cshrc File " on page 30)
# chmod 744 /export/skel/enduser/.*
```

▼ How to Start Admintool

1. **Verify that the following prerequisites are met. To use Admintool, you must:**
   - Have a bit-mapped display monitor. The Admintool software can be used only on a system with a console that is a bit-mapped screen such as a standard display monitor that comes with a Sun workstation.
   - Be running an X Window System, such as the OpenWindows environment.

- Be a member of the sysadmin group (group 14).

If you want to perform administration tasks on a system with an ASCII terminal as the console, use Solaris commands instead. See **useradd**(1M) for more information.

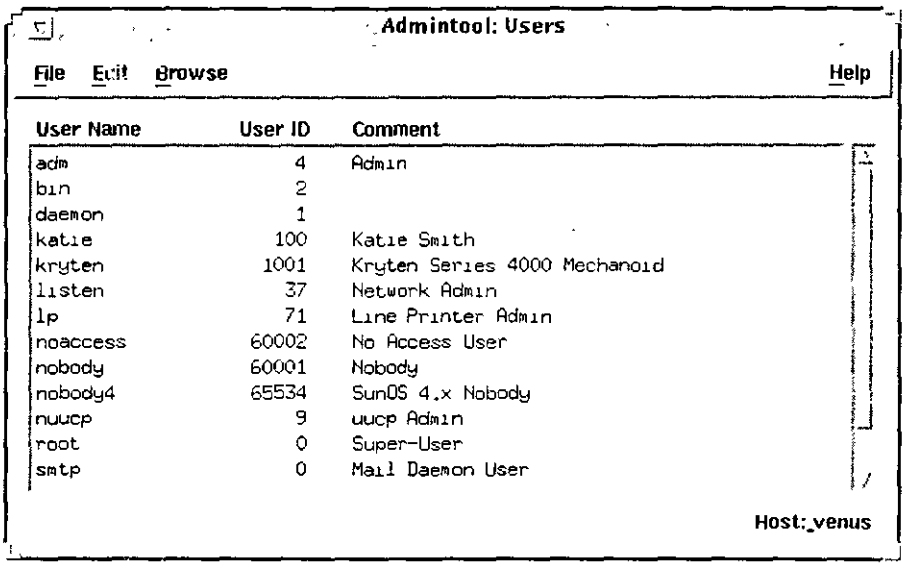
2. **Start Admintool.**

```
$ admintool &
```

The Users main window is displayed.

Example—Starting Admintool

This is the Users main window, which enables you to manage user account information.



▼ How to Add a Group

1. **Start Admintool, if it's not already running.**

   See "How to Start Admintool" on page 37 for more information on starting Admintool.

2. **Choose Groups from the Browse menu.**

   The Groups window is displayed.

3. **Select Add from the Edit menu.**

The Add window is displayed. If you need information to complete a field, click on the Help button to see field definitions for this window.

4. **Type the name of the new group in the Group Name text box.**

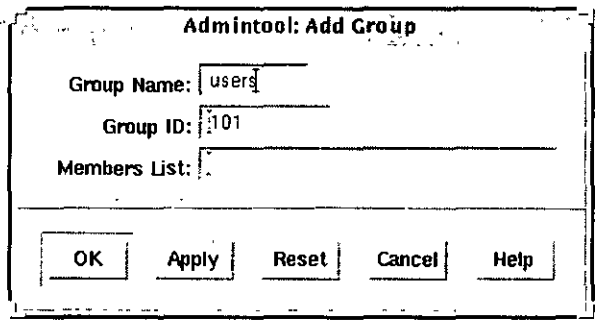
5. **Type the group ID for the new group in the Group ID text box.**
   The group ID should be unique.

6. **(Optional) Type user names in the Members List text box.**
   The list of users will be added to the group. User names must be separated by commas.

7. **Click on OK.**
   The list of groups displayed in the Groups window is updated to include the new group.

Example—Adding a Group

The following example adds a group named `users` that has a group ID of 101.



▼ How to Add a New User Account

1. **(Optional) Fill out the user information data sheet on "User Information Data Sheet" on page 35.**

2. **Start Admintool, if it's not already running.**
   See "How to Start Admintool" on page 37 for more information.

3. **Choose Add from the Edit menu.**
   The Add User window is displayed.

4. **Fill in the Add User window.**

If you need information to complete a field, click on the Help button to see field definitions for this window.

5. **Click on OK.**

   The list of user accounts displayed in the Users main window is updated to include the new user account.

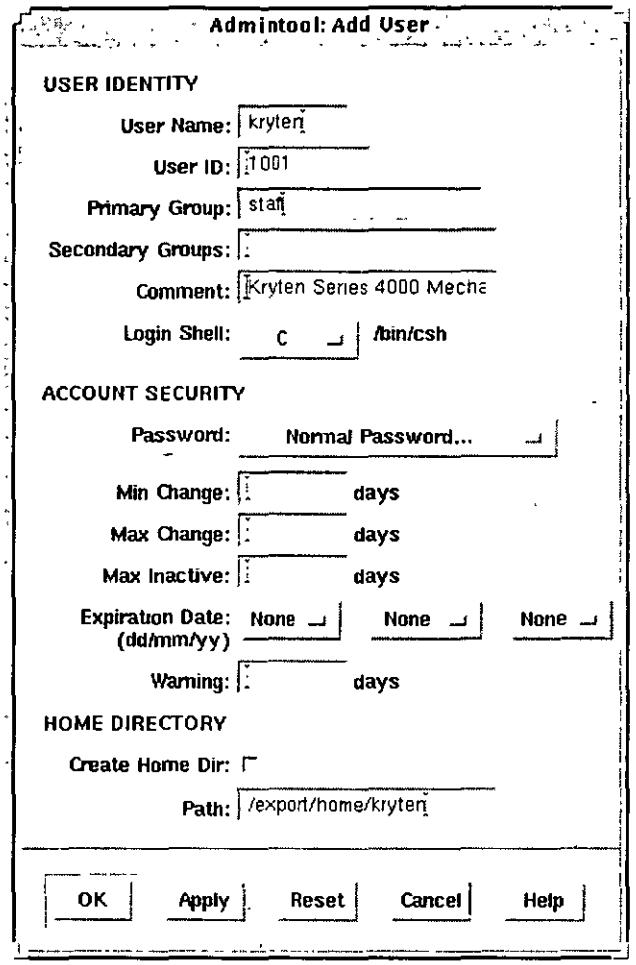
Where to Go From Here

If you created a user's home directory, you must share the directory so the user's system can remotely mount it. See "How to Share a User's Home Directory" on page 41 for detailed instructions.

If disk space is limited, you can set up a disk quota for the user in the file system containing the user's home directory. See "Managing Quotas (Tasks)" in *System Administration Guide, Volume II* for information on setting disk quotas.

Example—Adding a New User Account

The following example adds the user `kryten` to the system.



▼ How to Share a User's Home Directory

1. Become superuser on the system that contains the home directory.
2. Verify that the `mountd` daemon is running.

```
# ps -ef | grep mountd
root   176     1  0   May 02 ?        0:19 /usr/lib/nfs/mountd
```

The `/usr/lib/nfs/mountd` line is displayed if the `mountd` daemon is running.

3. If the `mountd` daemon is not running, start it.

```
# /etc/init.d/nfs.server start
```

4. List the file systems that are shared on the system.

```
# share
```

5. Determine your next step based on whether the file system containing the user's home directory is already shared.

If the File System Containing the User's Home Directory Is ...	Then ...
Already shared	Go to the verification step below.
Not shared	Go to Step 6 on page 42

6. Edit the `/etc/dfs/dfstab` file and add the following line.

```
share -F nfs /file-system
```

file-system Is the file system containing the user's home directory that you need to share. By convention, the file system is `/export/home`.

7. Share the file systems listed in the `/etc/dfs/dfstab` file.

```
# shareall -F nfs
```

This command executes all the `share` commands in the `/etc/dfs/dfstab` file, so you do not have to wait to reboot the system.

8. Verify that a user's home directory is shared, as follows:

```
# share
```

Where to Go From Here

If the user's home directory is not located on the user's system, you have to mount the user's home directory from the system where it is located. See "How to Mount a User's Home Directory" on page 43 for detailed instructions.

Example—Sharing a User's Home Directory

```
# ps -ef | grep mountd
# /etc/init.d/nfs.server start
# share
# vi /etc/dfs/dfstab

(The line share -F nfs /export/home is added.)

# shareall -F nfs
# share
-               /usr/dist                 ro   ""
-               /export/home/user-name    rw   ""
```

▼ How to Mount a User's Home Directory

1. **Make sure that the user's home directory is shared. See "How to Share a User's Home Directory" on page 41 for more information.**

2. **Log in as superuser on the user's system.**

3. **Edit the `/etc/vfstab` file and create an entry for the user's home directory.**

```
system-name:/export/home/user-name - /export/home/user-name nfs - yes rw
```

system-name	Is the name of the system where the home directory is located.
`/export/home/`*user-name*	Is the name of the user's home directory that will be shared. By convention, `/export/home` contains user's home directories; however, this could be a different file system
-	Are required placeholders in the entry.
`/export/home/`*user-name*	Is the name of the directory where the user's home directory will be mounted

See Chapter 28 for more information about adding an entry to the `/etc/vfstab` file.

4. **Create the mount point for the user's home directory.**

```
# mkdir -p /export/home/user-name
```

5. **Mount the user's home directory.**

```
# mountall
```

All entries in the current `vfstab` file (whose `mount at boot` fields are set to `yes`) are mounted.

6. **Use the `mount` command to verify that the home directory is mounted.**

Example—Mounting a User's Home Directory

```
# vi /etc/vfstab

(The line  venus:/export/home/ripley - /export/home/ripley
nfs - yes rw is added)

# mkdir -p /export/home/ripley
# mountall
# mount
/ on /dev/dsk/c0t3d0s0 read/write/setuid/
largefiles on Tue Jun 2 12:37:36 1998
/usr on /dev/dsk/c0t3d0s6 read/write/setuid/
largefiles on Tue Jun 2 12:37:36 1998
/proc on /proc read/write/setuid on Tue Jun 2 12:37:36 1998
/dev/fd on fd read/write/setuid on Tue Jun 2 12:37 38 1998
/opt on /dev/dsk/c0t3d0s5 setuid/read/write/
largefiles on Tue Jun 2 12:37:38 1998
/tmp on swap read/write on Jun 2 12:37:39 1998
/export/home/ripley on venus:/export/home/ripley /read/write/
remote on Jun 2 12:37:40 ...
```

Maintaining User Accounts Task Map

TABLE 2–2 Task Map: Maintaining User Accounts

Task	Description	For Instructions, Go To
1. Modify a Group	Modify a group's name or the users in a group by choosing Modify from the Edit menu in the Groups window.	"How to Modify a Group" on page 45
2. Delete a Group	Delete a group by choosing Delete from the Edit menu in the Groups window.	"How to Delete a Group" on page 46
3. Modify a User Account	*Disable a User Account* If you want to temporarily disable a user account, lock the user account from the Password menu in the Modify window. *Change a User's Password* If you want to change a user's password, use the Password menu in the Modify window. *Change Password Aging* If you want to force users to change their passwords periodically, change the Password Aging fields in the Modify window (Account Security category).	"How to Disable a User Account " on page 48 "How to Change a User's Password " on page 49 "How to Change Password Aging for a User Account " on page 50
4. Delete a User Account	Delete a user account by choosing Delete from in the Edit menu in the Users window.	"How to Delete a User Account " on page 51

▼ How to Modify a Group

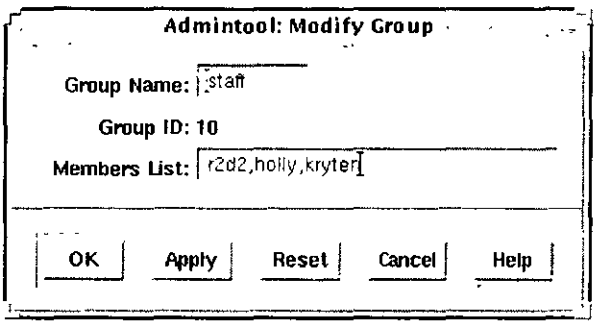
1. **Start Admintool, if its not already running. Select Groups from the Browse menu.**
   See "How to Start Admintool" on page 37 for more information.
2. **Select the group entry you want to modify from the Groups window.**
3. **Choose Modify from the Edit menu.**
   The Modify Group window is displayed containing the selected group entry.
4. **Modify either the group's name or the users in the group.**

User names must be separated by commas. If you need information to complete a field, click on the Help button to see field definitions for this window.

5. **Click on OK.**
   The group information displayed in the Groups window is updated.

Example—Modifying a Group

The following example adds the users `r2d2`, `holly`, and `kryten` to the `staff` group.



▼ How to Delete a Group

1. **Start Admintool, if it's not already running. Select Groups from the Browse menu.**
   See "How to Start Admintool" on page 37 for more information.

2. **Select the group entry you want to delete from Groups window.**

3. **Choose Delete from the Edit menu.**
   A window is displayed asking you to confirm the deletion.

4. **Click on OK.**
   The group entry is deleted from the Groups window.

▼ How to Modify a User Account

1. **Start Admintool, if it's not already running. Select Users from the Browse menu.**
   See "How to Start Admintool" on page 37 for more information

2. **Select the user account entry to modify from the Users window.**

3. **Choose Modify User from the Edit menu.**

   The Modify window is displayed containing the selected user account entry.

4. **Modify the user account.**

   If you need information to complete a field, click on the Help button to see field definitions for this window. You can change any of the Account Security fields, which includes changing a password or changing password aging. See the following tasks for detailed step-by-step instructions:

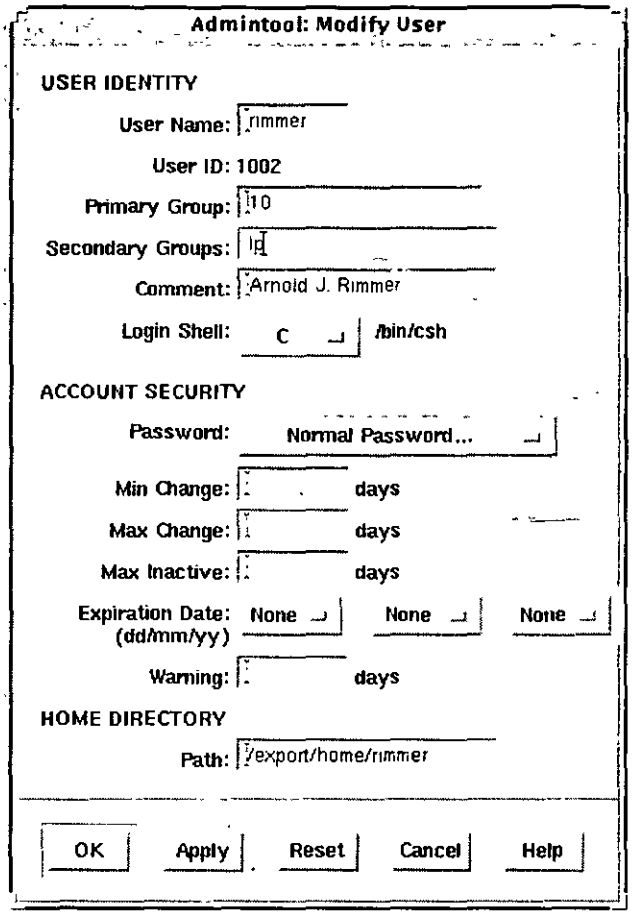
   - "How to Disable a User Account " on page 48
   - "How to Change a User's Password " on page 49
   - "How to Change Password Aging for a User Account " on page 50

5. **Click on OK.**

6. **To verify that the modifications were made, double-click on the modified user account entry in the Users window, then click on Cancel to close the window without making any modifications.**

Example—Modifying a User Account

The following example adds the secondary group membership `lp` to the `rimmer` user account.



▼ How to Disable a User Account

Note - You can enable the user account by changing the password status to Normal Password or Cleared Until First Login.

---

1. Start Admintool, if it's not already running. Select Users from the Browse menu, if necessary.

   See "How to Start Admintool" on page 37 for more information

2. Select the user account entry to be disabled.

3. Choose Modify from the Edit menu.

   The Modify Users window is displayed containing the selected user account entry.

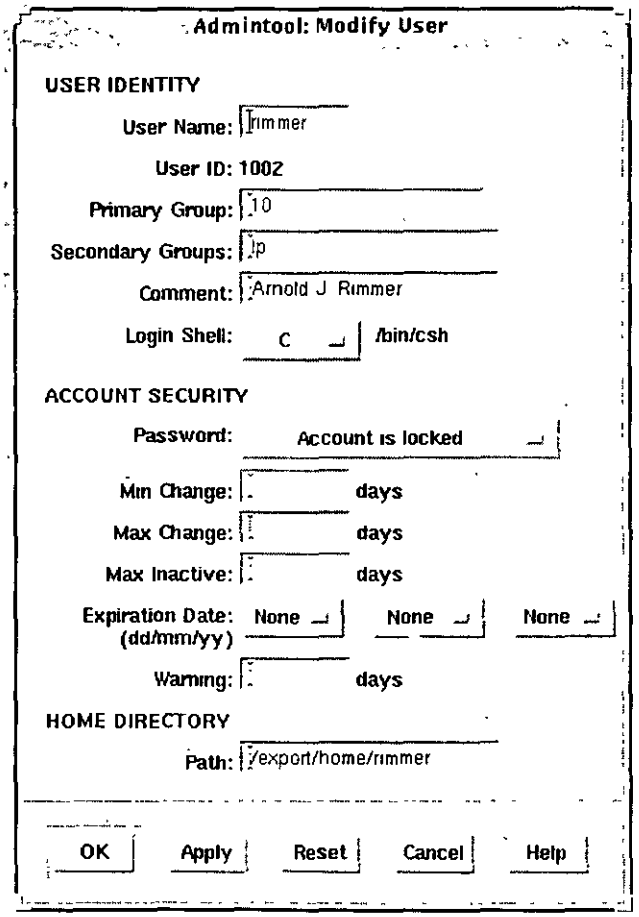
4. **Choose Account Is Locked from the Password menu.**
   This selects the locked password status, which disables the user account.

5. **Click on OK.**

6. **Verify that you have disabled the user account by attempting to log in with the disabled user account.**

Example—Disabling a User Account

The following example disables the `rimmer` user account.



▼ How to Change a User's Password

1. **Start Admintool, if it's not already running. Select Users from the Browse menu.**

See "How to Start Admintool" on page 37 for more information.

2. **Select the user account entry that needs the password changed.**

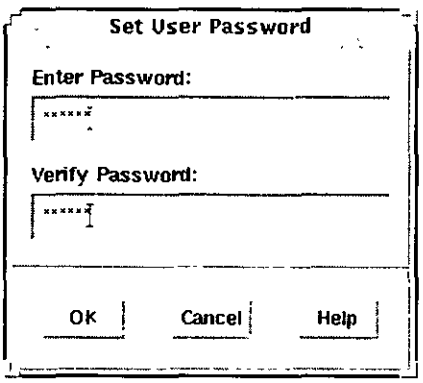
3. **Choose Modify from the Edit menu.**
   The Modify User window is displayed containing the selected user account entry.

4. **Choose Normal Password from the Password menu.**

5. **Click on OK.**

Example—Changing a User's Password

This is the pop-up window used to change user's passwords that is available from the Add User or Modify User windows.



▼ How to Change Password Aging for a User Account

1. **Start Admintool, if its not already running. Select Users from the Browse menu.**
   See "How to Start Admintool" on page 37 for more information.

2. **Select the user account entry that needs its password aging changed.**

3. **Choose Modify from the Edit menu.**
   The Modify window is displayed containing the selected user account entry.

4. **Change the following fields that affect password aging:**
   - Min Change

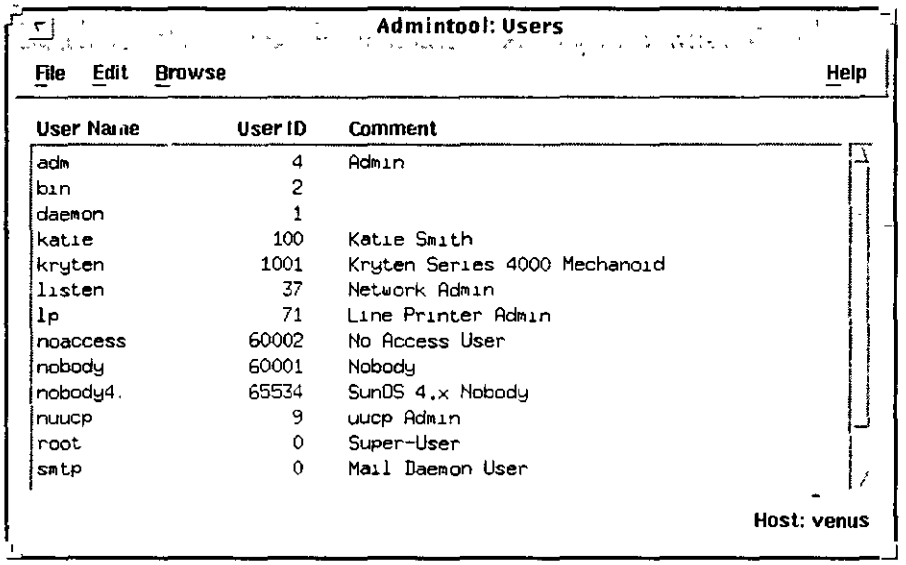
- Max Change
- Max Inactive
- Expiration Date
- Warning

If you need information about the password aging fields that are part of the Account Security category, click on the Help button.

5. **Click on OK.**

Example—Changing Password Aging for a User Account

In the following example, the user must keep a new password for at least one day (`Min Change`), and must change the password every 60 days (`Max Change`). The user must change the password if the account is inactive for more than 10 days (`Max Inactive`).



▼ How to Delete a User Account

1. **Start Admintool, if it's not already running. Select Users from the Browse menu, if necessary.**

   See "How to Start Admintool" on page 37 for more information.

2. **Select the user account entry to remove from the Users window.**

3. **Choose Delete from the Edit menu.**

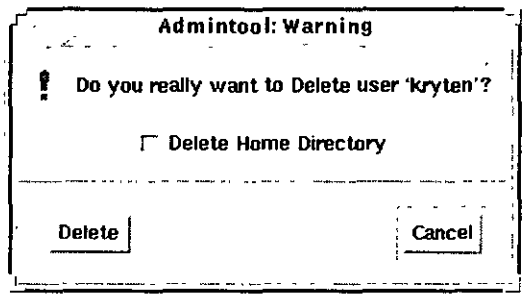
The Delete window is displayed to confirm the removal of the user account.

4. **(Optional) Click on the check box to delete the user's home directory and its contents.**

5. **Click on OK when you are ready to delete the user account. The user account entry is deleted from the Users main window.**

Example—Deleting a User Account

The account for user `kryten` and the `/export/home/kryten` directory is removed.



Solaris User Registration

Solaris User Registration is a tool for getting information about new Solaris releases, upgrade offers, and promotions. This graphical user interface (GUI) automatically starts when you first log into your desktop. The GUI lets you register now, later, or never. The registration process also provides Sun with the user's Solaris version, survey type, platform, hardware, and locale

Accessing Solaris™ Solve™

Completing the Solaris User Registration process provides access to Solaris Solve, an exclusive web site that offers valuable Solaris product information and solutions—all in one convenient location. It provides a quick and easy method for getting the most recent information on what's happening around the latest Solaris release. Solaris Solve also provides a preview to additional Sun contract and service opportunities.

Basically, the steps for completing Solaris User Registration and accessing Solaris Solve are:

1. Fill in the electronic Solaris User Registration profile.

2. Submit the profile by email or print the profile to fax or mail.
3. Create your login ID and password to access the Solaris Solve site.

   Even if you do not access the Solaris Solve site immediately, we recommend that you create your Solaris Solve login ID and password during the Solaris User Registration process. A Solaris Solve login ID and password should contain 6 to 8 alphanumeric characters without spaces and colons.
4. Access the Solaris Solve site.

Note - Solaris User Registration is not invoked if the system administrator or user is logged in as superuser.

If you choose to register, a copy of the completed form is stored in `$HOME/.solregis/uprops`. If you choose to never register and change your mind later, you can start User Registeration by:

- Typing `/usr/dt/bin/solregis` at any command line prompt, or
- Clicking the Registration icon in the Application Manager's desktop tools folder (Common Desktop Environment desktop only)

See **`solregis`**(1) for more information.

Troubleshooting Solaris User Registration Problems

This section provides troubleshooting tips for solving Solaris User Registration problems.

The following table describes problems that may occur when you try to register, and actions required to resolve these conflicts.

TABLE 2-3 Registration Problem Descriptions and Suggested Resolutions

Problem Description	How to Resolve the Problem
The registration form failed to initialize. Web page window displays and requests user see system administrator to resolve problem that prevents registration setup.	Check for missing registration files.
The form could not be emailed. Dialog box displays and requests user see system administrator to resolve problem.	Check to see if email is configured correctly. Also check if CDE is on user's system since it must be present to email completed registration form. Alternatively, users can print the form and fax or mail it.

TABLE 2–3 Registration Problem Descriptions and Suggested Resolutions *(continued)*

Problem Description	How to Resolve the Problem
The form could not be printed: Dialog box displays and requests user to see system administrator to resolve problem.	Check to see if the printer is configured correctly. Alternatively, the user can email form
The form could not be saved: Dialog box displays and verifies that registration succeeded; however, the registration information cannot be recalled when updating registration in the future.	Check the user's home directory. Required action depends on the system's configuration.
You forgot your Solaris Solve login ID and password.	Send a mail message describing the problem to `SolarisSolve@sun.com` or see "How to Restart Solaris User Registration" on page 54
You want to restart the registration process.	"How to Restart Solaris User Registration" on page 54

▼ How to Restart Solaris User Registration

Use the following procedure to restart the Solaris User Registration process.

1. **Change to the `$HOME/.solregis` directory.**

```
% cd $HOME/.solregis
```

2. **Remove the `uprops` file.**

```
% rm uprops
```

3. **Restart the registration process.**

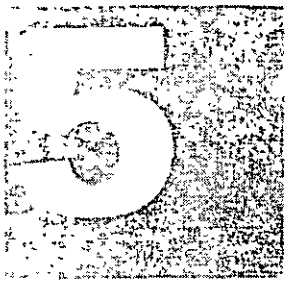
```
% /usr/dt/bin/solregis &
```

▼ How To Disable User Registration

Table 2–4 shows how to disable User Registration before and after installing Solaris software. Before disabling Solaris User Registration, Sun recommends that system administrators register for their organization.

TABLE 2–4 Ways to Disable User Registration

To Disable User Registration ...	You Can ...	For More Information See ...
Before Solaris software is installed	■ Deselect the `SUNWsregu` package (interactive installation) ■ Modify a custom JumpStart profile to not install the `SUNWsregu` package ■ Create and run a finish script that creates a file named `solregis` in the `/etc/default` directory on one or more systems with the following line in it: `DISABLE=1`	*Solaris Advanced Installation Guide* **`solregis`**(1)
After Solaris software is installed	■ Use the `pkgrm` command to remove the `SUNWsregu` package ■ Add the `solregis` file in the `/etc/default` directory (custom JumpStart installation only)	Chapter 17 *Solaris Advanced Installation Guide* **`solregis`**(1)



INTRODUCCIÓN A UNIX

SHELL

SHELL

Introducción a la programación con shell (Shell scripts).

Una gran parte del uso del sistema Unix consiste en emitir órdenes. Cuando usted emite una orden está relacionándose con el shell, la parte del sistema Unix a través de la cual se controlan los recursos del sistema operativo Unix. El shell proporciona muchas de las características que hacen al sistema Unix un entorno potente y Flexible. Se trata de un interprete de ordenes, un lenguaje de programación y más. Como interprete de ordenes, el shell interpreta las ordenes que se introducen y dispone de lo necesario para que se ejecuten. Además se puede utilizar el lenguaje de ordenes del shell como un lenguaje de programación de alto nivel para crear grandes programas denominados guiones.

Dentro de las diferentes versiones de Unix, para diferentes plataformas, se han desarrollado una serie de shells con características especificas. El *Bourne shell* es el más antiguo. Lo escribió Stephen Bourne en Bell Laboratories. Ha sido el estándar de facto del mercado Unix. Carece de las funciones interactivas o construcciones programáticas complejas que poseen el *Korn* shell o el *C* shell.

El *C shell* fue escrito por Bill Joy en la Universidad de Berkeley. Su sintaxis es muy parecida a la del lenguaje *C*, teniendo comandos e instrucciones de control similares. Otra característica significativa es el seguimiento y gestión de la historia de comandos ejecutados.

El *Korn shell* lo escribió David Korn, en Bell Laboratories y es compatible con casi todas las funciones del *C shell* y *Bourne shell*. Tiene las mismas facilidades interactivas que el *C shell*, pero es más rápida, y podemos editar la línea de comandos (*doskey* del DOS).

El *Key shell* fue desarrollado por HP y establece una interface sobre el *Korn shell*. Contiene menús y temas de ayuda en línea que ofrecen directrices sobre como ejecutar tareas tales como: editar, imprimir, listar, etc. Puede ser modificada para incluir nuestras propias utilidades y temas de ayuda.

Para conocer en que *shell* nos encontramos trabajando, ejecutamos echo $SHELL, comando que nos mostrará el valor de la variable de entorno SHELL, que contiene el nombre del programa *shell* que estamos ejecutando en ese momento.

A no ser que nos hallemos en un *shell* restringido (*rsh*, *rksh*), podemos cambiar de *shell* temporalmente. Este nuevo *shell* se invoca tecleando el nombre del shell que queremos ejecutar; en ese momento, el prompt cambiará de acuerdo al shell escogido. Una vez en el nuevo *shell*, ejecutaremos exit o [CTRL D], cuando queramos cambiar al *shell* original.

Si el cambio queremos efectuarlo permanentemente, ejecutaremos chsh <*usuario*> <*path_del_shell*>(*change shell*). Estos cambios serán validos en el momento de entrar a una nueva sesión del usuario.
Los comandos del *shell* admiten generalmente opciones y parámetros. Las opciones van precedidas generalmente por un signo -, y van separados del comando, otras opciones o parámetros, por un espacio o varios. Los parámetros, o variables, son datos que el comando necesita para ejecutarse de una forma determinada.

El propósito de este capítulo es brindarte alguna idea de las funciones disponibles a través de los shells y de sus funciones generales. Detalles de la programación de shell son discutidas en otra clase, "UNIX Bourne Shell Programing".

Unix hace uso completo del juego de caracteres ASCII. A diferencia de los comandos de lenguajes de sistemas operativos como VMS o NOS, UNIX es más sensitivo. En adición , varios caracteres tienen significados especiales para el shell. Nosotros ya hemos visto que el slash (/) para el shell indica el directorio raiz, y es usado con directorios, subdirectorios, y nombre de archivos para indicar un pathname absoluto y relativo.
Otros caracteres especiales que tienen significado para shell son:
` ' $ { } || && ;

Un comando de entrada es usualmente tomado del teclado, y un comando de salida es normalmente mostrado en el monitor. Una entrada por el teclado es referida como una "entrada estándar" o "stdin", y una salida por pantalla es referida como una "salida estándar" o "stdout".

REDIRECCIONAMIENTO DE ENTRADA

Es posible indicar a UNIX que obtenga datos de un archivo que de el teclado. Esto es llamado redireccionamiento de entrada. Para indicar que un comando de entrada viene de un archivo que de el teclado, se usará el caracter de redireccionamiento de entrada (<).
comando < archivo-de-entrada
comando= comando de shell.
archivo de entrada= es el archivo que tendrá la entrada para la ejecución del comando.

Un truco de memoria:

El símbolo menor que actúa como un embudo. Si tu vacías agua en la parte ancha, esta fluirá por la parte angosta. El archivo de entrada vacía su contenido en el comando.

EJEMPLO:

$mail neri < report

El archivo llamado reporte será enviado al login neri. Mail normalmente espera que la entrada venga de una entrada estándar, el teclado. El símbolo de redireccionamiento causa que la entrada al mail venga de un archivo llamado report.

REDIRECCIONAMIENTO DE SALIDA

También es posible indicar a UNIX que envíe los datos a un archivo, en lugar de enviarlo por default al monitor. Esto es llamado redireccionamiento de salida. Para indicar que la salida de un comando se guarde en un archivo en vez de que sea desplegado en el monitor, se usará el caracter de redireccionamiento de salida (>).

comando > archivo-destino-salida
comando= comando de shell.

archivo-destino-salida= archivo que recibirá la salida proveniente del comando.

El truco de memoria continua trabajando; solo que ahora el embudo indica hacia el archivo que recibirá la salida.

EJEMPLO:

ls -l > listing

La salida del comando ls no será desplegado en la pantalla, en su lugar esta salida estará en el archivo llamado listing. Si el archivo no existe, el shell lo creará. Si este ya existe, este archivo será sobrescrito, borrando la información anterior.

CUIDADO:

El shell no mostrará ninguna advertencia acerca de la sobreescritura del archivo original.

REDIRECCIONAMIENTO DE SALIDA ADJUNTO

El siguiente comando de shell también podrá redireccionar la salida a un archivo, pero en lugar de sobreescribir el archivo existente, este adjuntará la salida al final del archivo de salida.
comando >> archivo-de-salida
comando: un comando de shell.
archivo-de-salida: es el que recibirá la salida del comando.

Créanlo o no, el truco de memoria sigue trabajando; únicamente en este caso, un embudo alimenta en otro. Es decir la salida es vertida al final de archivo de salida.

EJEMPLO:

$ls -l >> listing

La salida del comando ls aparecerá en el archivo listing, sin destruir ningún dato existente. Si el archivo no existe, el shell lo creará.

REDIRECCIONAMIENTO DE ENTRADA Y SALIDA

El redireccionamiento de entrada y salida puede ocurrir en la misma línea de comando.
comando < archivo-de-entrada > archivo-de-salida
comando: un comando de shell
archivo-de-entrada: archivo que suplirá la entrada para la ejecución del comando.
archivo-de-salida: recibe la salida del comando.

EJEMPLO:

```
$cat pedro
Esta es una carta para Pedro.
$cat alicia
Esta es una carta para Alicia.
$cat linda
Esta es una carta para Linda.
$cat pedro alicia linda > todos
$cat todos
Esta es una carta para Pedro.
Esta es una carta para Alicia.
Esta es una carta para Linda.
$
```

Los primeros tres mandatos despliegan el contenido de tres archivos, pedro, alicia y linda. El mandato siguiente muestra a cat con tres nombres de archivo como argumentos.Cuando se da a cat más de un nombre de archivo, copia los archivos, uno a la vez, en su salida estándar. En este caso, la salida estándar se redirecciona al archivo todos, y todos recibe la concatenación de los tres archivos, como muestra el mandato final.

La técnica siguiente es útil cuando se desea realizar el mismo cambio en varios archivos. Elaborando un archivo de mandatos de editor que realizan el cambio requerido, y redireccionamiento después la entrada al editor para que provenga de ese archivo, puede ahorrarse el tiempo y la molestia de editar los archivos individualmente.

EJEMPLO:

```
$cat > cambio
$/carta/nota/
w
q
```

<CONTROL-D>
$

En la parte anterior se redirecciona la entrada a ed para cambiar la palabra carta por nota en el archivo alicia. En el caso de utilizar ed para hacer el cambio, se considera qué mandatos es necesario introducir una vez que se llama a ed; se meten esos mandatos en un archivo, y se ejecuta ed en el archivo de texto, empleando entrada que es redireccionada para proceder del archivo de mandatos.
La parte anterior muestra a cat creando un archivo de mandatos cambio. cambio contiene justo los caracteres que deben introducirse en el teclado para hacer que ed cambie la palabra carta por nota. Para llevar a cabo la sustitución, se llama a ed, y se la da un mandato de sustitución para realizar el cambio, un mandato w para grabar el archivo modificado y, por último un mandato q para dejar de usar el editor.

```
$cat alicia
Esta es una carta para Alicia.
$de alicia > cambio
30
31
$cat alicia
Esta es una nota para Alicia.
$de pedro < cambio > /dev/null
$cat pedro
Esta es una nota para Pedro.
$
```

Esta parte muestra una sesión de edición que utiliza entrada redireccionada. ed edita el archivo llamado alicia. En vez de recibir mandatos del teclado, éstos se toman del archivo llamado cambio. Los dos números que aparecen después de llamar a ed son la única salida que ed genera: el número de caracteres que de lee y escribe. Sólo se ha redireccionado la entrada a ed. La salida sigue yendo hacia la terminal. Si a un mandato s le sigue un mandato p, en el archivo cambio la salida generada por p aparecerá entre los números.

La segunda parte muestra el redireccionamiento de la entrada y la salida ed. Se edita un archivo llamado pedro con la entrada que proviene de cambio. La salida se redirecciona a /dev/null, un archivo nulo. Siempre puede enviarse una salida indeseada a /dev/null y el sistema la descartará.

CONECTORES (PIPES)

La salida de un comando puede ser usada como la entrada de un segundo comando, por medio del simbolo pipe (|), sin utilizar ningún archivo temporal. En algunas terminales el simbolo pipe es una barra vertical y en otras es una barra vertical partida

por la mitad. Ambas trabajaran exactamente igual. El siguiente formato muestra como usar el comando pipe:
comando1 | comando2

EJEMPLO:

$man acct | pg

La salida del comando man es procesada por el comando pg antes de aparecer en la pantalla. Normalmente la salida del comando man aparecerá en el monitor línea después de línea hasta que llega al final de archivo. En este caso, la salida es conectada al comando pg; y la pantalla se ira deteniendo cada 23 líneas, para que puedes leer la información.

COMODINES

Los comodines son caracteres especiales que provoca que el shell busque en un rango de posibles valores.
? representa cualquier caracter, mientras
* representa cualquier número de caracteres incluyendo ninguno.

EJEMPLO:

jo?eph

Esto indica que la tercer letra de la cadena "jo eph" debe ser cualquier caracter simple. Cualquier caracter puede ser sustituido por el caracter ?, incluyendo caracteres numéricos y especiales.
Para limitar un rango de posibles valores, encierra las posibilidades en corchetes.

EJEMPLO:

jo[a-z]eph

Este ejemplo limita el rango de caracteres en un conjunto de caracteres definido de a hasta la z. Los caracteres numéricos, o caracteres especiales no estarán dentro del partido.

Usando una coma como separador entre las opciones, nosotros podremos hacer mas restrictivo el rango.

EJEMPLO:

jo[s,m,5]eph

El único juego de caracteres que podrán ser elegidos son s, m y el número 5. Ningún otro caracter podrá ser utilizado.

El string jos*, provoca que el shell busque todo string que comience con las letras jos, mientras que [i-k]*h encontrará todo string que comience con "y", "j", o "k" y termine con "h".

Los comodines son extremadamente usado en gran variedad de aplicaciones. Por ejemplo, si tu quieres usar las paginas de man (manual), pero no conoces el nombre exacto de un comando dentro del sistema de contabilidad (accounting), podrás intentar con lo siguiente:
$man c*ac

Todos los comandos que inicien con las letras acc seguidas por cualquier string (incluyendo ninguno) serán pasadas al comando man como argumentos.
Si tú quieres obtener una lista de todos los archivos que terminen en .c (esta es la terminación de los programas hechos en C), dentro de tu directorio de trabajo; podrás teclear el siguiente comando:

EJEMPLO:

$ls *.c

Si queremos que el shell detenga la interpretación de un caracter especial, este deberá ir precedido del backslash (/) o encerrado entre comillas simples.

EJEMPLO:

jo/?eph
o
`jo?eph'

Ambos ejemplos representan al sting jo?eph. El shell no interpretará el **carácter** ? como un comodín.

RESTABLECIENDO UN TRABAJO BACKGROUND

Los procesos en UNIX pueden correr en la forma foreground o background. Los procesos foreground son interactivos; la entrada es leída por el teclado o por una entrada estándar, y la salida de desplegará en pantalla o en una salida estándar. Los procesos background corren sin interactuar con alguna terminal interactiva. Un proceso interactivo puede ser suspendido tecleando el caracter break desde el prompt de shell.

EJEMPLO:

$ctrl Z
suspended

El comando jobs despliega información sobre todas las sesiones de trabajo. El trabajo más reciente es marcado con un signo de más (+), y el que sigue de este es marcado con un guión o con un signo de menos (-). Un trabajo inicia cuando ejecutas cualquier comando.
El número de trabajos permitidos por usuario es determinado por el administrador de la red. El número de trabajos pueden ser de 1 a 16 con un default de 3.
El comando para desplegar la información sobre los trabajos concurrentes es:
$jobs

Si no hay trabajos, aparecerá de nuevo el prompt, y si hay algunos trabajos suspendidos aparecerá como sigue:

EJEMPLO:

$jobs
+1 rlogin domax1
-2 rlogin domax1
$

Esto muestra que hay dos trabajos suspendidos. Ambos trabajos hicieron una conexión remota a domax1. Esto es únicamente de ejemplo.

El comando fg (foreground) regresa un trabajo que estaba suspendido. El comando despliega el número de trabajo que se le asignó. Cuando no se dan argumentos, fg regresará el trabajo más reciente. Con un argumento numérico, fg regresa el trabajo específico.

Para conectarte a un trabajo suspendido (sesión) teclea el siguiente comando:
$ fg [n]
n = no. del trabajo foreground.
$fg
mandará el trabajo suspendido más reciente.

NAME

vi, view, vedit – screen-oriented (visual) display editor based on ex

SYNOPSIS

vi [– | **–s**] [**–l**] [**–L**] [**–R**] [**–r** [*filename*]] [**–t** *tag*] [**–v**] [**–V**] [**–x**] [**–w***n*] [**–C**] [**+***command* | **–c** *command*] *filename*...

view [– | **–s**] [**–l**] [**–L**] [**–R**] [**–r** [*filename*]] [**–t** *tag*] [**–v**] [**–V**] [**–x**] [**–w***n*] [**–C**] [**+***command* | **–c** *command*] *filename*...

vedit [– | **–s**] [**–l**] [**–L**] [**–R**] [**–r** [*filename*]] [**–t** *tag*] [**–v**] [**–V**] [**–x**] [**–w***n*] [**–C**] [**+***command* | **–c** *command*] *filename*...

AVAILABILITY

SUNWcsu

DESCRIPTION

vi (visual) is a display-oriented text editor based on an underlying line editor **ex**. It is possible to use the command mode of **ex** from within **vi** and vice-versa. The visual commands are described on this manual page; how to set options (like automatically numbering lines and automatically starting a new output line when you type carriage return) and all **ex** line editor commands are described on the **ex**(1) manual page.

When using **vi**, changes you make to the file are reflected in what you see on your terminal screen. The position of the cursor on the screen indicates the position within the file.

OPTIONS

Invocation Options

The following invocation options are interpreted by **vi** (previously documented options are discussed in the NOTES section of this manual page):

– | **–s**	Suppress all interactive user feedback. This is useful when processing editor scripts.
–l	Set up for editing LISP programs.
–L	List the name of all files saved as the result of an editor or system crash.
–R	**Readonly** mode; the **readonly** flag is set, preventing accidental overwriting of the file.
–r *filename*	Edit *filename* after an editor or system crash. (Recovers the version of *filename* that was in the buffer when the crash occurred.)
–t *tag*	Edit the file containing the *tag* and position the editor at its definition.
–v	Start up in display editing state using **vi**. You can achieve the same effect by simply typing the **–vi** command itself.
–V	Verbose. Any non-tty input will be echoed on standard error. This may be useful when processing editor commands within shell scripts.
–x	Encryption option; when used, **vi** simulates the **X** command of **ex** and prompts the user for a key. This key is used to encrypt and decrypt text using the algorithm of the **crypt** command. The **X** command makes an educated guess to determine whether text read in is encrypted or not. The temporary buffer file is encrypted also, using a transformed version of the key typed in for the **–x** option.

–w*n* Set the default window size to *n*. This is useful when using the editor over a slow speed line.

–C Encryption option; same as the **–x** option, except that **vi** simulates the **C** command of **ex**. The **C** command is like the **X** command of **ex**, except that all text read in is assumed to have been encrypted.

+command | **–c** *command*
Begin editing by executing the specified editor *command* (usually a search or positioning command).

The *filename* argument indicates one or more files to be edited.

The *view* invocation is the same as **vi** except that the **readonly** flag is set.

The *vedit* invocation is intended for beginners. It is the same as **vi** except that the **report** flag is set to 1, the **showmode** and **novice** flags are set, and **magic** is turned off. These defaults make it easier to learn how to use **vi**

vi Modes

Command	Normal and initial mode. Other modes return to command mode upon completion. ESC (escape) is used to cancel a partial command.
Input	Entered by setting any of the following options: **a A i I o O c C s S R**. Arbitrary text may then be entered. Input mode is normally terminated with ESC character, or, abnormally, with an interrupt.
Last line	Reading input for : / ? or !; terminate by typing a carriage return; an interrupt cancels termination.

COMMAND SUMMARY

In the descriptions, CR stands for carriage return and ESC stands for the escape key.

Sample commands

← ↓ ↑ →	arrow keys move the cursor
h j k l	same as arrow keys
i*text*ESC	insert *text*
cw*new*ESC	change word to *new*
easESC	pluralize word (end of word; append **s**; escape from input state)
x	delete a character
dw	delete a word
dd	delete a line
3dd	delete 3 lines
u	undo previous change
ZZ	exit **vi**, saving changes
:q!CR	quit, discarding changes
*/text*CR	search for *text*
^U ^D	scroll up or down
:*cmd*CR	any **ex** or **ed** command

Counts before vi commands

Numbers may be typed as a prefix to some commands. They are interpreted in one of these ways.

line/column number **z G |**

	scroll amount	^D ^U
	repeat effect	most of the rest
Interrupting, canceling	**ESC**	end insert or incomplete cmd
	DEL	(delete or rubout) interrupts
File manipulation	**ZZ**	if file modified, write and exit; otherwise, exit
	:wCR	write back changes
	:w!CR	forced write, if permission originally not valid
	:qCR	quit
	:q!CR	quit, discard changes
	:e *name*CR	edit file *name*
	:e!CR	reedit, discard changes
	:e + *name*CR	edit, starting at end
	:e +*n*CR	edit starting at line *n*
	:e #CR	edit alternate file
	:e! #CR	edit alternate file, discard changes
	:w *name*CR	write file *name*
	:w! *name*CR	overwrite file *name*
	:shCR	run shell, then return
	:! *cmd*CR	run *cmd*, then return
	:nCR	edit next file in arglist
	:n *args*CR	specify new arglist
	^G	show current file and line
	:ta *tag*CR	position cursor to *tag*

In general, any **ex** or **ed** command (such as *substitute* or *global*) may be typed, preceded by a colon and followed by a carriage return.

Positioning within file	**^F**	forward screen
	^B	backward screen
	^D	scroll down half screen
	^U	scroll up half screen
	*n*G	go to the beginning of the specified line (end default), where *n* is a line number
	/pat	next line matching *pat*
	?*pat*	previous line matching *pat*
	n	repeat last / or ? command
	N	reverse last / or ? command
	/pat/+n	nth line after *pat*
	?*pat***?−***n*	nth line before *pat*
	]]	next section/function
	[[	previous section/function
	(	beginning of sentence
	)	end of sentence
	{	beginning of paragraph

	}	end of paragraph
	%	find matching ( ) { or }
Adjusting the screen	^L	clear and redraw window
	^R	clear and redraw window if ^L is → key
	zCR	redraw screen with current line at top of window
	z–CR	redraw screen with current line at bottom of window
	z.CR	redraw screen with current line at center of window
	/*pat*/z–CR	move *pat* line to bottom of window
	z*n*.CR	use *n*-line window
	^E	scroll window down 1 line
	^Y	scroll window up 1 line
Marking and returning	``	move cursor to previous context
	''	move cursor to first non-white space in line
	m*x*	mark current position with the ASCII lower-case letter *x*
	`*x*	move cursor to mark *x*
	'*x*	move cursor to first non-white space in line marked by *x*
Line positioning	H	top line on screen
	L	last line on screen
	M	middle line on screen
	+	next line, at first non-white
	–	previous line, at first non-white
	CR	return, same as +
	↓ or j	next line, same column
	↑ or k	previous line, same column
Character positioning	^	first non white-space character
	0	beginning of line
	$	end of line
	l or →	forward
	h or ←	backward
	^H	same as ← (backspace)
	space	same as → (space bar)
	f*x*	find next *x*
	F*x*	find previous x
	t*x*	move to character prior to next *x*
	T*x*	move to character following previous *x*
	;	repeat last f, F, t, or T
	,	repeat inverse of last f, F, t, or T
	n|	move to column *n*
	%	find matching ({) or }

Words, sentences, paragraphs

w	forward a word
b	back a word
e	end of word
)	to next sentence
}	to next paragraph
(	back a sentence
{	back a paragraph
W	forward a blank-delimited word
B	back a blank-delimited word
E	end of a blank-delimited word

Corrections during insert

^H	erase last character (backspace)
^W	erase last word
erase	your erase character, same as **^H** (backspace)
kill	your kill character, erase this line of input
****	quotes your erase and kill characters
ESC	ends insertion, back to command mode
CTRL-C	interrupt, suspends insert mode
^D	backtab one character; reset left margin of *autoindent*
^^D	caret (^) followed by control-d (^D); backtab to beginning of line; do not reset left margin of *autoindent*
0^D	backtab to beginning of line; reset left margin of *autoindent*
^V	quote non-printable character

Insert and replace

a	append after cursor
A	append at end of line
i	insert before cursor
I	insert before first non-blank
o	open line below
O	open above
r*x*	replace single char with *x*
R*text***ESC**	replace characters

Operators

Operators are followed by a cursor motion, and affect all text that would have been moved over. For example, since **w** moves over a word, **dw** deletes the word that would be moved over. Double the operator, for example, **dd** to affect whole lines.

d	delete
c	change
y	yank lines to buffer
<	left shift
>	right shift
!	filter through command

Miscellaneous Operations

C	change rest of line (**c$**)
D	delete rest of line (**d$**)
s	substitute chars (**cl**)
S	substitute lines (**cc**)
J	join lines
x	delete characters (**dl**)
X	delete characters before cursor (**dh**)
Y	yank lines (**yy**)

Yank and Put

Put inserts the text most recently deleted or yanked; however, if a buffer is named (using the ASCII lower-case letters a - z), the text in that buffer is put instead.

3yy	yank 3 lines
3yl	yank 3 characters
p	put back text after cursor
P	put back text before cursor
"*x*p	put from buffer *x*
"*x*y	yank to buffer *x*
"*x*d	delete into buffer *x*

Undo, Redo, Retrieve

u	undo last change
U	restore current line
.	repeat last change
"*d*p	retrieve *d*'th last delete

AUTHOR

vi and **ex** were developed by The University of California, Berkeley California, Computer Science Division, Department of Electrical Engineering and Computer Science.

ENVIRONMENT

If any of the LC_* variables (LC_CTYPE, LC_MESSAGES, LC_TIME, LC_COLLATE, LC_NUMERIC, and LC_MONETARY) (see **environ**(5)) are not set in the environment, the operational behavior of **vi** for each corresponding locale category is determined by the value of the LANG environment variable. If LC_ALL is set, its contents are used to override both the LANG and the other LC_* variables. If none of the above variables is set in the environment, the "C" (U.S. style) locale determines how **vi** behaves.

LC_CTYPE

Determines how **vi** handles characters. When LC_CTYPE is set to a valid value, **vi** can display and handle text and filenames containing valid characters for that locale **vi** can display and handle Extended Unix code (EUC) characters where any individual character can be 1, 2, or 3 bytes wide **vi** can also handle EUC characters of 1, 2, or more column widths In the "C" locale, only characters from ISO 8859-1 are valid.

LC_TIME

Determines how **vi** handles date and time formats. In the "C" locale, date and time handling follows the U.S. rules.

FILES

/tmp	default directory where temporary work files are placed; it can be changed using the **directory** option (see the **ex**(1) **set** command)
/usr/share/lib/terminfo/?/*	compiled terminal description database
/usr/lib/.COPYterm/?/*	subset of compiled terminal description database

SEE ALSO

intro(1), **ed**(1), **edit**(1), **ex**(1), **environ**(5)

Solaris Advanced User's Guide

NOTES

Two options, although they continue to be supported, have been replaced in the documentation by options that follow the Command Syntax Standard (see **intro**(1)). A **–r** option that is not followed with an option-argument has been replaced by **–L** and **+***command* has been replaced by **–c** *command*.

The message **file too large to recover with –r option** , which is seen when a file is loaded, indicates that the file can be edited and saved successfully, but if the editing session is lost, recovery of the file with the **–r** option will not be possible.

The encryption options are provided with the Security Administration Utilities package, which is available only in the United States.

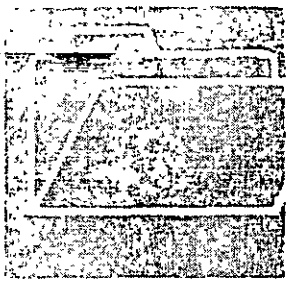
The editing environment defaults to certain configuration options. When an editing session is initiated, **vi** attempts to read the **EXINIT** environment variable. If it exists, the editor uses the values defined in **EXINIT**, otherwise the values set in **$HOME/.exrc** are used. If **$HOME/.exrc** does not exist, the default values are used.

To use a copy of **.exrc** located in the current directory other than **$HOME**, set the *exrc* option in **EXINIT** or **$HOME/.exrc**. Options set in **EXINIT** can be turned off in a local **.exrc** only if *exrc* is set in **EXINIT** or **$HOME/.exrc**.

Tampering with entries in **/usr/share/lib/terminfo/?/*** or **/usr/share/lib/terminfo/?/*** (for example, changing or removing an entry) can affect programs such as **vi** that expect the entry to be present and correct. In particular, removing the "dumb" terminal may cause unexpected problems.

Software tabs using ^T work only immediately after the *autoindent*.

Left and right shifts on intelligent terminals do not make use of insert and delete character operations in the terminal.



INTRODUCCIÓN A UNIX

ANEXO

BREVARIO HISTORICO

- Originado en los laboratorios BELL AT&T, antecesor del Sistema Operativo MULTICS finales de los 60´s
- KEN THOMPSON y DENNIS RITCHIE diseñadores originales constituyen un juego de viaje especial para la PDP-7
- Posteriormente crearon una nueva estructura de sistemas de archivos añadiendo entorno de procesos con planificación.

Notas:

BREVARIO HISTORICO

- Unix nace como una simplificación de MULTICS en 1970.
- En 1975 el proyecto es pasado a una máquina PDP-11
- Implementación original codificada en ensamblador.
- En 1971 primer cliente real fue la oficina de Abogados de patente BELL con el programa "TROFF".
- En 1973 el Kernell fue recodicado en "C"

Notas:

BREVARIO HISTORICO

- Entre 1970 y 1975 el sistema se desarrolla para máquinas superiores PDP-11/45 PDP-11/70.
- Provocando la venta de cientos de máquinas PDP-11
- las PDP-11 junto con UNIX se introducen fuertemente al mercado telefónico.
- Simultáneamente AT&T distribuye copias a muchas universidades del mercado.
- Se genera la versión BSD (Berkeley Software Distrubution) en la universidad de California de Berkeley

Notas:

BREVARIO HISTORICO

- AT&T fortalece a UNIX hacia la computación comercial.
- BSD domina en comunidades universitarias y técnicas.
- Comienza la competencia AT&T y BSD
- Finales de los 70´s AT&T comienza un nuevo esquema de nominación:
  - System III
  - Finales de los 80´s
  - System V
  - SVR2 y SVR3
  - System IV
  - Solo productos de transición.

Notas:

BREVARIO HISTORICO

- Finales de los 70´s y principios de los 80´s UNIX fue portado prácticamente a casi por todas las máquinas con potencial para soportarlo.
- 1986 se genera XENIX para equipos basados en el 8088 con participación de Microsoft
- 1989 Santa Cruz Operation libera su versión SCO UNIXSystem V
- 1993 Univel (USI y NOVELL) libera UNIXWARE

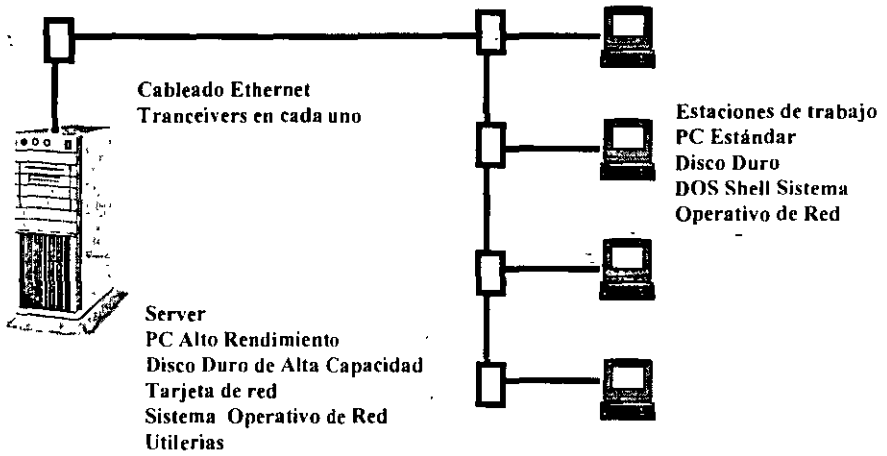
Notas:

SISTEMAS ABIERTOS

- "Abierto", debería ser lo opuesto a "propietario"
- Es una arquitectura de capas a interfaces bien definidas.
- Cada uno de los componentes puede evolucionar independientemente de los otros componentes con que se relacionen.

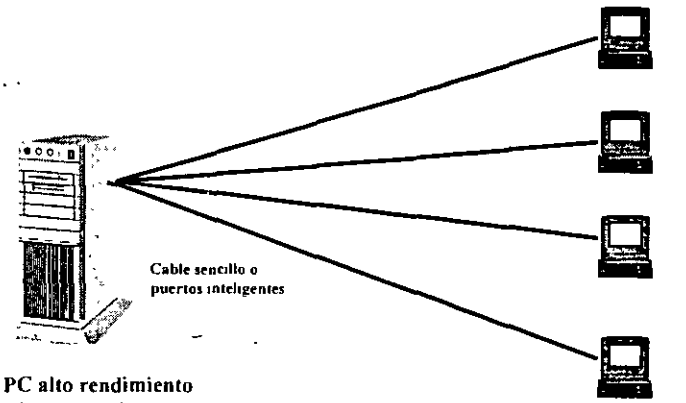
Notas:

CONFIGURACION DE UNA RED TIPICA.



Notas:

CONFIGURACION TIPICA DE UNIX

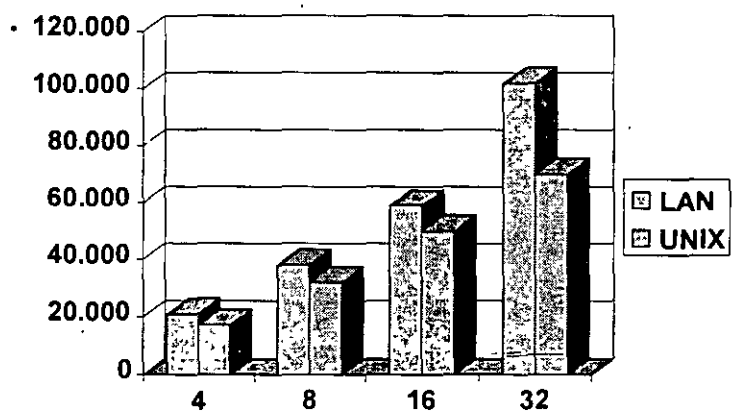


PC alto rendimiento
Disco Duro de alta capacidad
Tarjeta multipuertos
Sistema operativo UNIX

Terminales

Notas:

COSTOS APROXIMADOS POR SISTEMA POR NUMERO DE USUARIOS.

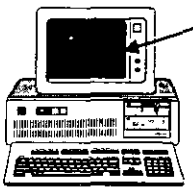


APUNTES:

Notas:

CONCEPTOS GENERALES

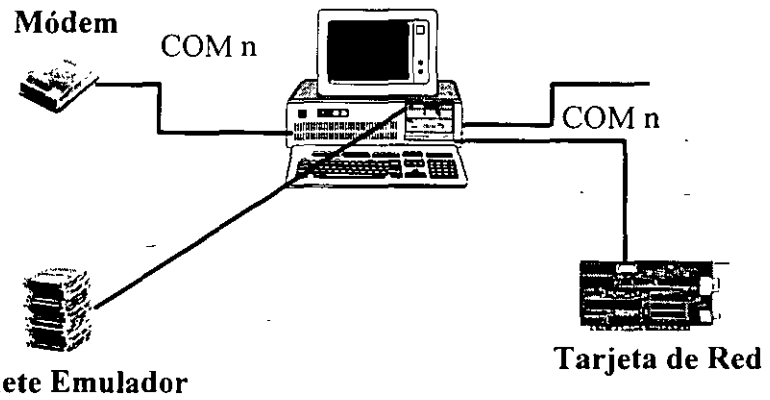
Terminales X



Notas:

CONCEPTOS GENERALES

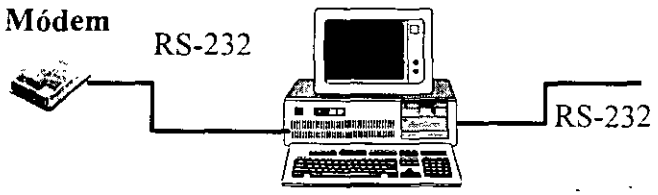
PC Emulando Terminal



Notas:

CONCEPTOS GENERALES

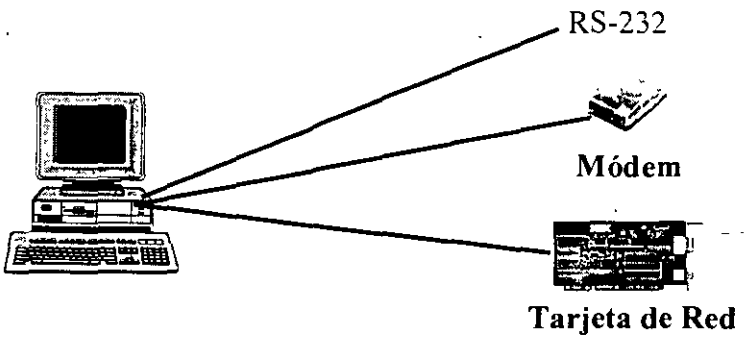
Terminal Texto



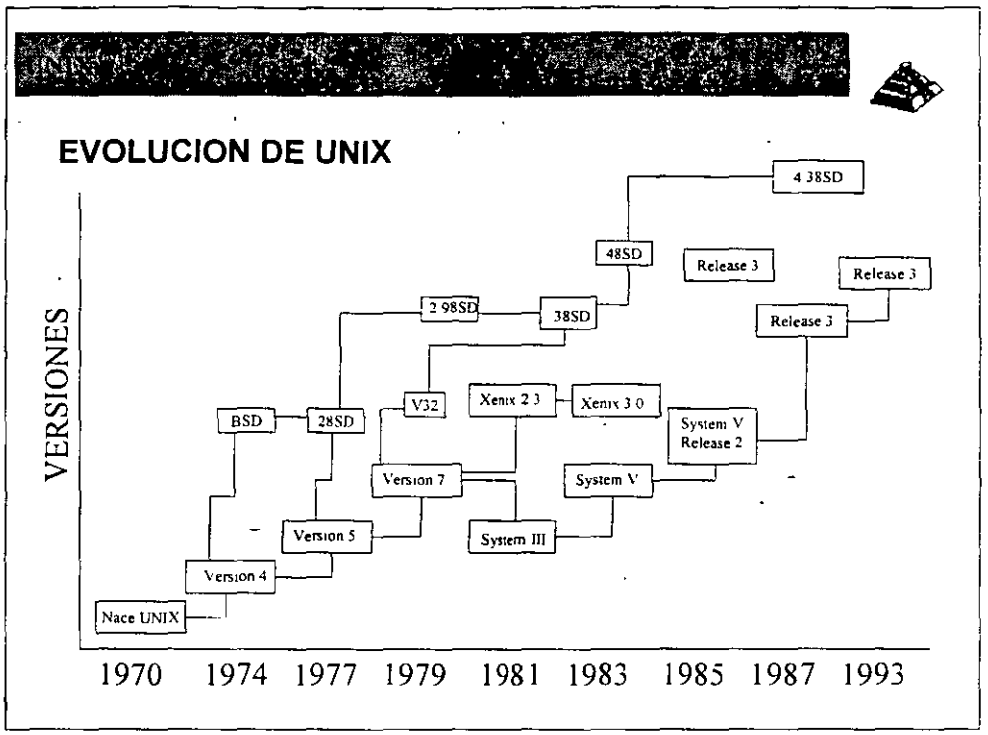
Notas:

CONCEPTOS GENERALES

PC Emulador X-Terminal



Notas:


EVOLUCION DE UNIX
VERSIONES
4 3BSD
4BSD
Release 3
Release 3
Release 3
2 9BSD
3BSD
BSD
2BSD
V32
Xenix 2 3
Xenix 3 0
System V
Release 2
Version 7
System V
Version 5
System III
Version 4
Nace UNIX
1970
1974
1977
1979
1981
1983
1985
1987
1993

Notas:

SISTEMAS ABIERTOS

- No existe una definición absoluta
- Corre bajo UNIX
- Se adecua a las Normas Internacionales
- Tienden a evolucionar
- Son capaces de Integrar.

Notas:

UNIX PROTAGONISTAS IMORTANTES

• AT&T	UNIX SYSTEM V
• HEWLETT-PACKARD	HP/UX
• IBM	AIX
• NEXT	NEXT STEP
• OSF(Open Software Foundation)	OSF/1
• SANTA CRUZ OPERATION	SCO UNIX
• SUN SOFT	SOLARIS
• UNIVEL (USL&NOVELL)	UNIXWARE

- HP,IBM,DEC,APOLO,SIEMANS,NIXDOFF,GRUPO BULL

Notas:

UNIX EN RED

- Los servicios que proporcionan los servidores UNIX
  - Aplicaciones de base de datos (principalmente SQL)
  - Comunicaciones TCP/IP, X.25, monitoreo de la red, etc.
  - Aplicaciones "X" gráficas
  - Servicios de archivos e impresoras para clientes DOS
  - Aplicaciones UNIX

Notas:

UNIX EN RED

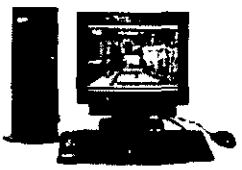
Las ventajas para los usuarios :

- Poder correr aplicaciones de DOS simultáneamente con aplicaciones UNIX de carácter.
- Colectividad. A una Red Ethernet (LAN o WAN) con TCP/P se pueden conectar:
  - Terminales tontas
  - PCs
  - "Workstation"
  - Mainframes
- Heterogeneidad de marcas
- Simetria.Los clientes puede lanzar aplicaciones de cualquier servidor puede ser también cliente
- Sistemas abiertos

Notas:

WORKSTATION

- Estaciones de Trabajo



Notas:

WORKSTATION

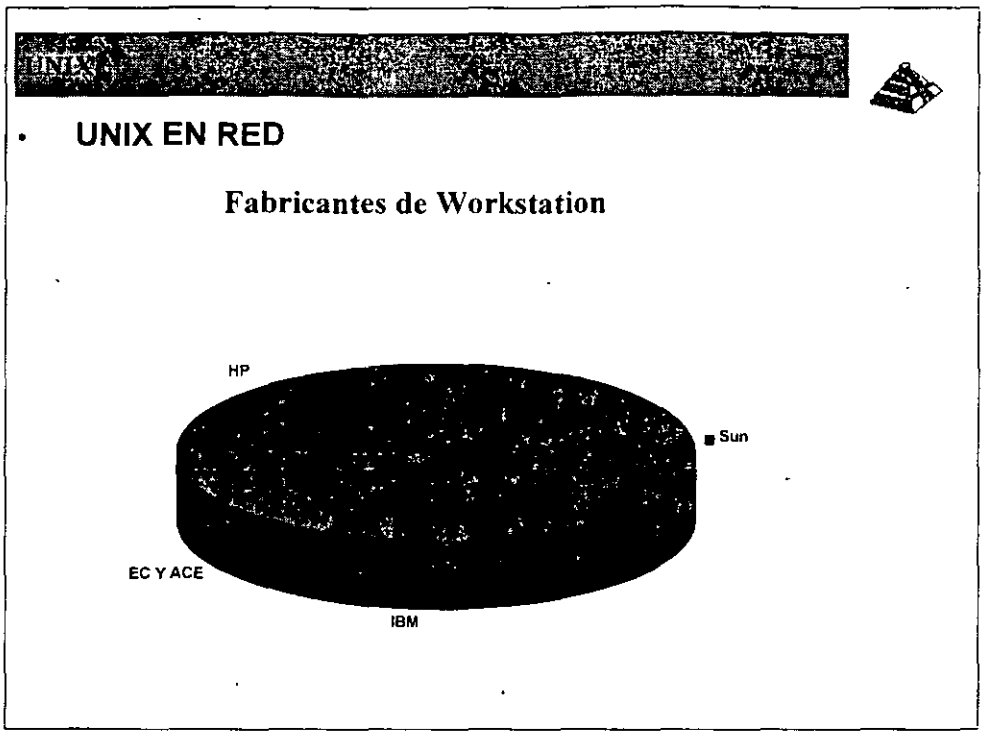
- 1.-Procesador poderoso basado en tecnología RISC
- 2.-Pantalla gráfica y ratón (mouse)
- 3.-Tarjeta Ethernet o Token-ring integrada
- 4.-Sistema operativo UNIX con ventanas X11
- 5.- Grupos de trabajo

Notas:

UNIX EN RED

- FABRICANTES DE WORKSTATION
  - Sun Microsystems
  - Hewlet Packard
  - Digital Equipment
  - IBM
  - Otros

Notas:


UNIX
• UNIX EN RED
Fabricantes de Workstation
HP
Sun
EC Y ACE
IBM

Notas:

UNIX EN RED

- PRODUCTOS DE SOFTWARE D.O.S. PARA CONECTIVIDAD UNIX

PRODUCTO	COMPAÑÍA
XOFTWARE FOR DOS	AGE LOGIC
JSB MULTIVIEW	
PC-INTERFACE	LOCUS
TCP/IP PARA DOS	LOCUS
PC XSIGHT	

Notas:

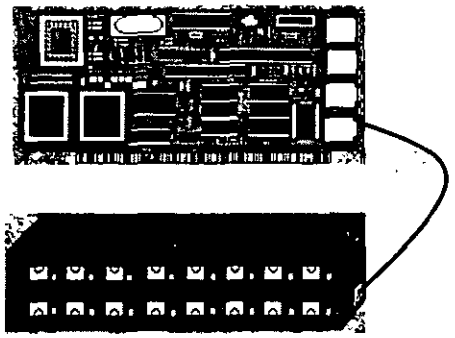
UNIX

HARDAWARE DE UNIX EN RED

Notas:

UNIX HARDWARE PARA REDES

- TARJETA MULTIPUERTOS



Notas:

SERVIDOR DE TERMINALES

- ¿QUÉ ES?
- Estos dispositivos tienen la posibilidad de conectar de 8 a 255 puertos seriales a una red EtHernet.
- Cuentan con soporte de protocolos TCP/IP y LAT de DEC.

Notas:

SERVIDOR DE TERMINALES

- CARACTERISTICAS PRINCIPALES
  - Procesador: Intel 80386 a 16 Mhz
  - Número de puertos: 8 a 255
  - Distancias: hasta 914m.
  - Memoria: hasta 512 Kb
  - Soporte a protocolos: TCP/IP y LAT de DEC
  - Conectores Ethernet: BNC, AUI, 10 Base T
  - Cuenta con un puerto paralelo centronics.

Notas:

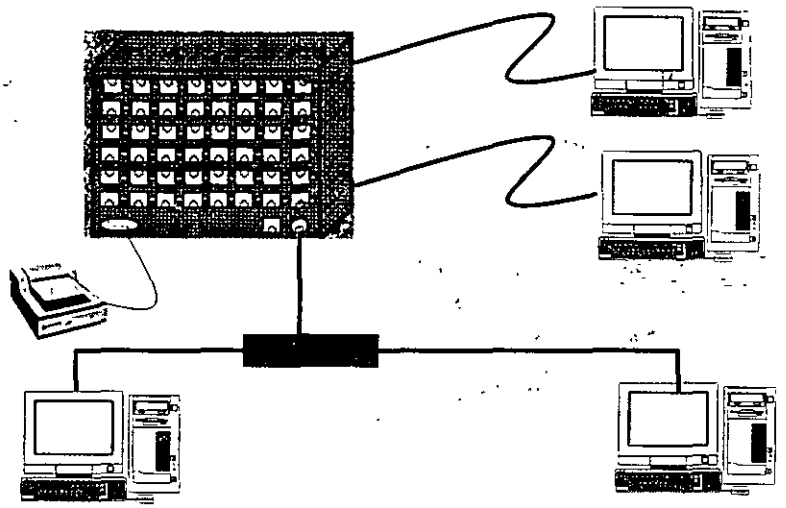
TARJETA MULTIPUERTOS

PRINCIPALES CARACTERISTICAS

- No inteligente
- inteligente
- procesadores: RISC de 10 Mhz a 16 Mhz.
- Intel 80186 de 10 Mhz a 16 Mhz.
- Memoria: de 64 Kb a 512Kb.
- Arquitectura: ISA, EISA, MCA, RS6000
- Número de puertos : de 4 a 96
- Velocidad: de 75bps a 38,400 Bps
- Con o sin soporte a Módem.

Notas:

SERVIDOR DE TERMINALES



Notas: