

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

INSTALACION Y MANEJO DE REDES (LAN)

EN PLATAFORMA UNIX

MODULO III DEL DIPLOMADO

MATERIAL DIDACTICO

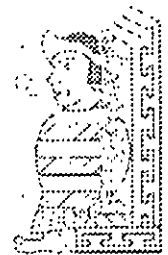
MAYO DE 1996

2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025



INSTALACIÓN Y MANEJO DE REDES (LAN) EN PLATAFORMA UNIX

1.- INTRODUCCIÓN A UNIX Y TCP/IP



Mayo de 1996.

TEMARIO MODULO III (ux)

1.- INTRODUCCIÓN A UNIX Y TCP/IP

- Conceptos generales
- Antecedentes de Unix
- Unix en plataformas Intel
- Unix en otras plataformas

2.- UNIX EN RED.

- Características en ambiente de Red
- Enlaces Unix - Ethernet
- Enlaces Unix - Token Ring
- Enlaces Puerto Serial
- Hardware de Unix para Red

3.- FABRICANTES DE UNIX.

- Unix System V de SCO
- Solaris de Sun Microsystems
- Unixware de Univel (Novell & USL)
- Otros

4.- INSTALACIÓN DE UNIX

- Instalación del Hardware
- Instalación Plataforma base Unix
- Instalación del Sistema Operativo de Red
- Configuración y administración de Protocolos

5.- ADMINISTRACIÓN DE UNIX EN RED

- El kernel, Generalidades y configuración
- Swapping
- Sistemas de seguridad y Filesystem
- Utilerías de administración
- Usuarios, Grupos de usuarios, Altas, Bajas, cambios, etc.
- Atributos y Seguridades
- Instalaciones de aplicaciones

6.- AMBIENTE GRÁFICO UNIX

- Conceptos de ODT
- Manejo de ODT
- Terminales X-Windows
- Emulación X-Terminal

7.- INTEGRACIÓN DE UNIX CON OTROS SISTEMAS

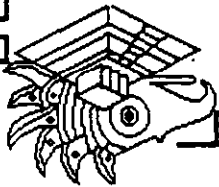
- Introducción a TCP/IP
- Unix, Netware, Lan Manager, Lmx, DOS

8.- CONECTIVIDAD

- Enlaces locales
- Enlaces remotos

9.- SESIONES DE TALLER EN CADA PUNTO DEL TEMARIO

1.- INTRODUCCION A
UNIX Y TCP/IP



M. J.

HISTORIA DEL SISTEMA UNIX

Unix se originó en los Laboratorios Bell AT&T, una de las instituciones de investigación mejor dotadas de los Estados Unidos, su historia es casi única en comparación con otros sistemas operativos debido a que los avances son en gran parte aportaciones de personas con ideas creativas singulares. La implicación es que los avances no han venido principalmente de decisiones burocráticas sino más bien directamente de las necesidades y creatividad de los usuarios. Esto sigue siendo cierto hoy, lo que hace del sistema UNIX uno de los jardines más fértiles para la creación de nuevos conceptos en computación. El sistema UNIX fue diseñado por un grupo de personas que eran representantes de AT&T en el desarrollo de una de las influencias germinales en la computación moderna, el Sistema Operativo MULTICS, desarrollado en MIT a finales de los sesenta.

Como uno de los primeros sistemas de tiempo compartido MULTICS incorporó la mayoría de las ideas que aparecen en los sistemas multitarea actuales. Desgraciadamente, MULTICS sufrió las consecuencias de su papel innovador y resultó mucho más complejo y pesado de lo que era necesario. A finales de los sesenta AT&T abandonó la mayor parte de su participación, en el proyecto MULTICS, dejando a un grupo de personas con talento pero frustradas, con muchas ideas acerca de lo que un sistema en tiempo compartido debería ser.

Sin acceso al sistema MULTICS, estas personas se quedaron sin un Sistema Operativo moderno con el cual trabajar, de modo que crearon uno nuevo. Los diseñadores Ken Thompson y Dennis Ritchie construyeron el sistema basado en un diseño elaborado con Rudd Canaday. Pronto se les unieron J.F. Ossana y R. Morris. Tras un período de discusiones, adquirieron una computadora DEC PDP-7 de desecho y se pusieron a trabajar. Como muchos de los mejores proyectos, éste comenzó con la creación de un juego. Thompson y Ritchie desarrollaron un juego de viaje espacial para la PDP-7.

Después de esta experiencia crearon una nueva estructura de sistema de archivos y un nuevo software que es muy similar al sistema de archivos modernos. Le añadieron un entorno de procesos con planificación y completaron el resto de un Sistema Operativo rudimentario. El nombre UNIX pronto se aplicó a los resultados ya que su trabajo fue una simplificación del sistema MULTICS. El sistema estuvo operando sobre el PDP-7 a principios de 1970, y a mediados de esa década habían pasado el proyecto a una máquina DEC PDP-11 de reciente aparición.

Muchas de las ideas claves del sistema UNIX moderno estaban presentes en las primeras implementaciones, incluyendo el sistema de archivos, la implementación de procesos y la estructura de las líneas de orden aún utilizadas hoy en día.



La implementación original fue codificada en lenguaje ensamblador, pero pronto se desarrolló el lenguaje de programación C dentro del grupo, empezando en 1971. El lenguaje C fue utilizado casi inmediatamente en la continuación del desarrollo del sistema UNIX, y en 1973 el núcleo se recodificó en C. Hoy sólo unas cuantas subrutinas del núcleo de alto rendimiento están escritas en lenguaje ensamblador. Este fue el primer intento de codificar un Sistema Operativo entero en un lenguaje de alto nivel y la portabilidad que se le consiguió está ampliamente considerada como una de las razones principales de la popularidad que el sistema UNIX actualmente goza.

Al mismo tiempo se iniciaron las herramientas de proceso de textos que posteriormente dieron lugar a *troff*, y el primer cliente real del sistema UNIX fue la Oficina de Abogados de Patentes de los Laboratorios Bell, que empezó a utilizar el programa *troff* en otoño de 1971.

El sistema UNIX captó inmediatamente la imaginación de los informáticos en los Laboratorios Bell y después de dos o tres años había alrededor de una docena de sistemas UNIX ejecutándose en varias máquinas diferentes. Se realizaron con frecuencia importantes mejoras software y AT&T comenzó a soportar el sistema como producto interno dentro de los Laboratorios Bell. El programa *troff* apareció durante este período, entre muchas otras innovaciones.

Sin embargo, el sistema UNIX adquirió cuerpo con el desarrollo de las máquinas PDP-11 superiores, tales como la PDP-11/45 y la PDP-11/70, entre principios y mediados de los setenta. El sistema UNIX se ajustaba de forma natural a la arquitectura DEC y ocasionó la venta de muchos cientos de máquinas PDP-11 a lo largo de los años. Los programadores dentro de los Laboratorios Bell empezaron a utilizar máquinas UNIX para su trabajo de procesado de textos, y los diseñadores de productos de Sistemas Bell comenzaron a utilizar PDP-11 con sistemas UNIX para sistemas llave en mano dentro del negocio telefónico.

Simultáneamente, AT&T remitió muchas copias del sistema UNIX a todas las universidades del mundo, y una generación completa de informáticos a finales de los setenta aprendió su profesión con el sistema UNIX. Esto dio lugar a otra fértil ola de innovaciones y la implementación ampliamente utilizada BSD (Berkeley Software Distribution) apareció en la Universidad de California en Berkeley. Al tiempo que AT&T fortalecía el sistema UNIX y lo optimizaba en la dirección de la computación comercial, las versiones BSD resultaban dominantes en las comunidades universitarias y técnicas.

La compatibilidad entre las versiones BSD se presentan equiparadas con las versiones AT&T, aunque los equipos en ambos lados se apresuran a incorporar las mejores innovaciones del otro sistema a sus propias versiones.



A finales de los setenta, AT&T comenzó un nuevo esquema de nominación para su versión del sistema UNIX. Anteriormente las versiones principales se designaban según las nuevas versiones que salían del área de investigación, y dos de las más populares fueron las denominadas Revisión Sexta y luego Revisión Séptima. Siguiendo una reorganización interna del soporte del sistema UNIX, AT&T cambió su numeración a Sistema III y Sistema V. Realmente estas nuevas versiones eran descendientes directas de la Revisión Séptima y el Sistema V suplantó al Sistema III a mediados de los ochenta. El Sistema IV fue utilizado internamente en los Laboratorios Bell, pero se consideró un producto de transición que nunca fue soportado públicamente.

A finales de los ochenta AT&T normalizó el nombre de Sistema V y sus versiones recientes se denominan Sistema V, Revisión 2 y Sistema V, Revisión 3, que a menudo se abrevian como SVR2, SVR3, respectivamente. Durante los últimos años setenta y los primeros ochenta, una o ambas de las versiones BSD y AT&T fueron portadas a casi todos los computadores con potencia para soportarlas.

Esto generalmente exigía como mínimo unidades de disco de alta velocidad y soporte de gestión de memoria interna en la CPU, aunque algunas versiones experimentales han sido adaptadas a máquinas basadas en ROM sin disco rígido en absoluto. Hoy en día se pueden comprar versiones del sistema UNIX para los mayores supercomputadores, las máquinas maxicomputadores más ampliamente utilizadas y casi todos los minicomputadores a la venta.

Conforme los microcomputadores se han desarrollado en velocidad y potencia y su costo ha disminuido, estas máquinas se han movido al rango del sistema UNIX. Las máquinas 8088 originales eran casi lo bastante potentes para soportar el sistema UNIX y algunas implementaciones podrían ejecutarse sobre estas máquinas. El Sistema operativo XENIX es una versión adelgazada del sistema UNIX para el IBM PC, pero está realmente en o por encima del filo de la capacidad de la máquina y sólo ha tomado cuerpo con las máquinas 80286 y 80386.

Recientemente, los Laboratorios Bell y AT&T han desarrollado una nueva versión genérica denominada Revisión Octava o sistema UNIX de Investigación. Aunque no se venda comercialmente, esta versión ha sido ampliamente distribuida a universidades.

Los descendientes de las versiones BSD están siendo constantemente mejorados y la realización de acuerdos entre AT&T y Microsoft, AT&T y Sun, y AT&T y Amdahl están permitiendo integrar más extensamente las versiones microcomputadores y supercomputadores. Finalmente se espera que las versiones SVR3, BSD y XENIX converjan en una versión única del sistema UNIX que pueda ejecutarse en casi cualquier entorno hardware. Este producto combinado podría también permitir la compatibilidad de código objeto entre diferentes versiones para la misma máquina.



LA REVISION SVR3

Es la versión más actualizada del sistema UNIX de AT&T. Ha sido portada a la mayoría de los principales computadores y es el estándar actual para la línea AT&T. Ha sido significativamente mejorada con respecto a versiones anteriores y contiene muchas modificaciones. Las principales modificaciones a nivel de usuario incluyen más ayuda en línea, herramientas de administración del sistema notablemente mejoradas (y simplificadas) y mayor resistencia al daño debido a caídas de tensión y otros daños inadvertidos.

A niveles inferiores del sistema UNIX, las modificaciones más importantes han sido el soporte para bibliotecas compartidas, un soporte de memoria virtual muy mejorado y nuevas herramientas para integrar redes de área local con el núcleo. Naturalmente ha habido muchísimos cambios y optimizaciones menores en todo el sistema.

SVR3 FRENTE A BSD Y SVR2

El sistema SVR3 está significativamente más libre de errores que las versiones BSD e incorpora muchas de las innovaciones que se originaron en los sistemas BSD. Sobre todo, hay mejor soporte para SVR3 que para las versiones BSD, las versiones BSD están fragmentándose en diferentes vendedores que mejoran el sistema por sus propios medios. Casi todos los sistemas comerciales utilizan SVR3, mientras los sistemas científicos y técnicos tienden a construirse a partir de la base BSD.

Comparado con su predecesor inmediato, la versión SVR2 de AT&T, SVR3 tiene varias características nuevas, pero es mayor y a menudo más lento. Es decir, SVR3 requiere significativamente más memoria real y un disco rígido mayor que SVR2. Por contra, el usuario de SVR3 obtiene un sistema avanzado con nuevas características de conexión a red, y mejor soporte de documentación y herramientas de administración.

BSD (BERKELEY SOFTWARE DISTRIBUTION)

En 1974, el campus Berkeley de la Universidad de California se involucró en el desarrollo del UNIX cuando el Profesor Fabry adquirió la versión 4. En 1975, Ken Thompson visitó la Universidad, su Alma Mater, y ayudó a instalar la versión 6 en una PDP-11/70. El mismo año, dos graduados llegaron a Berkeley: Bill Joy y Chuck Haley, los cuales tuvieron un papel determinante en el desarrollo del sistema. Ellos y Thompson trabajaron en un compilador en Pascal y un editor llamado ED, y posteriormente volcaron su interés en las operaciones internas del kernel, llamado aquí el nombre de Berkeley Software Distribution.



Posteriormente Bill Joy siguió trabajando sobre el EX para añadirle capacidad de direccionamiento de cursor sobre terminales CRT y producir además el *C-shell*, que se llamó así por su similitud con el ambiente de programación "C". En 1978 se actualizó la organización interna del sistema, llamándola Second Berkeley Distribution, que también se conoce como 2BSD.

El sistema se volvió popular en las máquinas PDP existiendo varios lanzamientos, hasta el 2.9BSD que aún en la actualidad se encuentra en algunas PDP-11. En el mismo año se adquirió una VAX-11/780 que inicialmente tenía el VMS de DEC. Sin embargo, el personal de investigación estaba ya habituado a trabajar en UNIX. Entonces el profesor Fateman obtuvo una copia de UNIX 32V, una versión 7, que se trasladó a la VAX. Bill Joy y otro graduado, Ozalp Babaoglu, adicionaron el manejo de memoria virtual al 32V, es decir, la posibilidad de correr programas de mayor tamaño que la memoria del equipo. Joy también trasladó las utilerías de la versión 2BSD a la VAX llamándola "Virtual VAX/UNIX".

En diciembre de 1979, este conjunto de modificaciones al 2BSD y a la versión 7 dieron origen al 3BSD. La Agencia de Proyectos Avanzados e Investigaciones de la Defensa (DARPA) aceptó el sistema para uso interno dando con esto el impulso necesario para que, tiempo después, se distribuyera la 4BSD. En 1983, la 4.2BSD incluía el Fast File System en el cual cada sistema de archivos se subdivide en un grupo de cilindros y a su vez el sistema operativo crea y graba archivos en cilindros paralelos.

Esto mantiene los sectores pertenecientes a un archivo en una misma región física del disco, evita así la fragmentación del mismo y permite un acceso más rápido. Esta versión soportaba la conexión de una red Ethernet. Tiempo después, Sun Microsystems le adicionó el Network File System (NFS). La liberación del 4.3BSD, en 1987, consistió en algunos ajustes a la 4.2BSD. Los cambios menos drásticos a este último lanzamiento, en contraste con los anteriores, han consistido en adiciones que indican que BSD y AT&T podrían converger eventualmente.

EL XENIX DE MICROSOFT

Xenix está basado en la versión 7 de AT&T. Microsoft liberó el Xenix 2.3 en 1980 como una implantación para microcomputadoras. De la misma manera que el sistema se basó en la versión 7, el Xenix tomó algunas utilerías de la 4.1BSD.

La versión 3.0 incorporó algunas características del AT&T System III y el Xenix 5.0 se diseñó tratando de cumplir con los estándares de la definición de interfaces de System V de AT&T. La intención de Microsoft y de Santa Cruz Operation, actualmente propietaria de Xenix, es lograr que Xenix y UNIX converjan en un sólo producto, es decir, SCO UNIX.



AT&T

Paradójicamente, AT&T no liberó formalmente su versión de UNIX hasta 1982, años después de que se distribuyeron el Xenix y la 4.1BSD. El primer lanzamiento comercial se llamó UNIX System III, que se basó principalmente en la versión 7 y en algunas características de programación de la versión 6. En 1983 se liberó el UNIX System V que incluía importantes utilerías de Berkeley. Se incorporó el proceso *init* de inicio de tareas, siendo diferente el procedimiento de la versión 7.

AT&T liberó el UNIX System V v.2 en 1984 introduciendo una versión propia de la base de datos Termcap, llamada Terminfo la cual consiste en una serie de archivos que describen las capacidades de cada modelo y tipo de terminal. Otros cambios incluyeron modificaciones menores al sistema jerárquico de archivos, la adición de Streams y el Remote File System en respuesta al NFS de Sun. El actual UNIX System V versión 3 (SVR3) es la correspondiente a las plataformas Intel y la base de los ambientes gráficos para UNIX.

EL FUTURO DE UNIX

El usuario puede confundirse ante la variedad de versiones, distintas marcas y hasta *clones*. Sin embargo, la gran corriente de la estandarización ha incluido al UNIX al crearse la SVID (System V Interface Definition; Definición de la Interface del System V) que norma con exactitud los servicios que el sistema operativo debe ejecutar y cómo deben solicitarse, además de las exigencias de diferentes organizaciones como IEEE, la DARPA y las propias asociaciones de usuarios que regulan todo cambio y adición.

Existen actualmente dos entidades que luchan por el liderazgo de los estándares, éstas son la OSF (Open Systems Foundation; Fundación de Sistemas Abiertos) y Unix International, las cuales agrupan a diferentes fabricantes de software y hardware.

Esta guerra por colocar en el mercado las primicias de la investigación, las mejores interfaces y los ambientes más amigables y prácticos traerá un solo ganador: el usuario.



SISTEMAS ABIERTOS

Los Sistemas Abiertos han llegado al mercado y ofrecen una perspectiva más al usuario para aprovechar al máximo el hardware y software con que cuenta. Del mismo modo las posibilidades de comercializar la nueva tecnología son más amplias para los distribuidores.

Todo esto es un atractivo adicional para los usuarios y una oportunidad para los desarrolladores propietarios Pero qué son los sistemas abiertos? Cómo saber cuando se está hablando de uno de ellos?

Los sistemas abiertos pueden caracterizarse como una tecnología orientada a la supervivencia para los 90, ya que representan una respuesta a las peticiones de la mayoría de usuarios activos que buscan el bienestar común.

La explicación anterior puede tomarse como sólo un rasgo de esta nueva tecnología, pues en realidad todavía no existe una definición absoluta que sea aceptable por el grueso de la población informática. Dentro de las definiciones más aceptadas con respecto a los sistemas abiertos, existen cuatro que han sido más o menos asimiladas.

- Los sistemas abiertos corren bajo UNIX.
- Se adecúan a las normas internacionales.
- Tienden a evolucionar.
- Son capaces de integrarse.

Los sistemas abiertos y UNIX, son utilizados por diversas organizaciones. Para unas, UNIX es el punto que marca la desaparición de las grandes computadoras en los procesos de operaciones comerciales. Para otras significa mucho más el remplazo de los sistemas operativos propietarios, tanto de los simples procesadores personales como de los complejos sistemas de cómputo.



Sin embargo, lo cierto es que, tratar de emparejar la tecnología de sistemas abiertos con el sistema Operativo UNIX, trae como consecuencia algunas limitantes. Como punto principal, es posible mencionar que todavía no hay una definición completamente estandarizada del sistema operativo UNIX. Además de que UNIX y su API (Application Programming Interface) no direccionan elementos claves de sistemas tales como "look" and "feel", manejo de información y desarrollo basado en sustitución. Por otra parte, las funciones comerciales más complicadas, son las que requieren de un sistema complejo que las soporte.

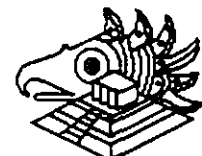
Los sistemas abiertos se acoplan a las normas internacionales, pero para evitar confusiones y antes de continuar, es indispensable aclarar que "abierto" debería ser lo opuesto de "propietario". Ser abierto es ser compatible. Lo cual hace de un sistema bajo este concepto, un elemento atractivo para convertir al equipo y al programa en productos compatibles.

Sin embargo, al respecto de esta sencilla y atinada definición, hay desacuerdo. Para empezar, una norma implica un acuerdo entre distribuidores y usuarios con el fin de que se suspenda la innovación en un área determinada, para que la creatividad e inventiva se canalicen en algún otro sector, evitando así la saturación de uno sólo.

De tal manera que cuando los desarrolladores han resuelto los problemas comerciales, puedan comenzar a promover las ventajas de las implantaciones basadas en normas, enfrentándolas a las nuevas alternativas propietarias. Otro inconveniente, es que lleva tiempo que usuarios y distribuidores coincidan en los movimientos normativos. Como consecuencia, las normas se direccionan a tecnologías antiguas en lugar de enfocarse al nivel de los líderes.

La definición de sistemas abiertos se puede describir mejor como una "terminación abierta", la cual se caracteriza por una arquitectura de capas e interfaces bien definidas donde cada uno de los componentes puede evolucionar independientemente de los otros componentes con que se relacionen. Por otra parte, mientras estos sistemas con terminación abierta, invitan a la exploración de una tecnología más avanzada y mejor, la asimilación de las normas puede verse como un avance con escalas o una carrera con obstáculos.

Esto no significa que haya un enfrentamiento entre las normas y una solución de determinación abierta. Lo cierto es que, las primeras protegen la inversión previniendo el "lock-in" propietario; mientras que la solución de determinación abierta protege la inversión permitiendo a la aplicación hacer uso de la nueva tecnología conforme ésta va surgiendo sin necesidad de gastar más haciendo más eficiente su equipo.



La definición de sistemas abiertos hace hincapié en la facilidad de combinar solución y/o componentes de diferentes fabricantes. Su susceptibilidad a integrarse proporciona protección de la inversión y una habilidad de innovación al poder combinar otros elementos, antiguos, existentes y mejorada tecnología a la vez que protege la inversión actual. El problema de esta definición es que la integración a lo largo de un eje no garantiza la integración a lo largo de otros ejes (por ejemplo: se puede tener una excelente integración de datos, pero contradicciones con otros componentes de la aplicación).

Dentro de las definiciones de la tecnología de sistemas abiertos más aceptadas, existe una gran ventaja: se toman en cuenta las necesidades de desarrollo y ambientes operacionales, que a su vez, proporcionan soluciones de aplicación al proteger la inversión en recursos humanos (operadores de cómputo, entrenamiento y usuarios esporádicos), equipo, aplicaciones y programas del sistema y datos. Además, responden en el acto a los cambios de concepto, conducción, escala y ubicación del negocio.

Siendo un tanto exagerados, un proteccionista es partidario de las normas, renuente a cambios lentos y bien pensados, se trata pues de un "conservador". En tanto que un "liberal" es partidario de los cambios y considera que las normas deben ser condenadas a la hoguera.

Ante esta situación antagónica, sería ideal hacer un balance entre los dos extremos. Podríamos decir, que esta es una llamada para actuar, para que los usuarios expresen a los distribuidores claramente sus necesidades y lo que esperan de la tecnología de los sistemas abiertos. La idea es que se emitan dos mensajes distintos: uno en cuanto a normas y en cuanto a innovación.

Con respecto a ésta última, será necesario que se continúe renovando, pero que no se cambie sólo por cambiar. Se debe estar plenamente seguro de que la innovación traera más beneficios que gastos, con la salida de las normas.

En cuanto a normas, es importante que se cumpla con las que describen el procedimiento que los programas de aplicaciones requieren para los servicios del sistema operativo (por ejemplo: que los API's estándares (Application Programming Interfaces) como el POSIX y las interfaces de servicio de presentación como MOTIF concurren y cumplan con las normas).



También se debe cumplir con las normas de comunicación que permiten la interconexión de diversos sistemas y con un modelo de información fuente (Repository Information Model) estandar. Esta norma deberá indicar el significado de los objetos que se guardan en la fuente y explicar cómo éstos se pueden acceder y manipular para describir o crear soluciones comerciales. Finalmente, este requerimiento remplazará la necesidad de APIs estándares, lenguajes e interfaces de programación; ya que las herramientas que popularizan y manipulan la fuente se convertirán en el medio de describir e implantar los sistemas.

Las normas para este modelo motivarán la innovación, permitiendo el desarrollo de nuevas herramientas y técnicas y el despliegue de los activos de la aplicación existentes representados en la fuente. Todas estas normas necesitan recibir la aprobación de múltiples distribuidores de equipo y programas.

El progreso de varias organizaciones normativas es lento, cómo se las arreglará un usuario mientras tanto?, Cuándo deberá aceptar las normas y cuándo emplear tecnología propietaria? Cómo formarse una idea absoluta entre las normas y la innovación? Cada situación busca ser juzgada y desgraciadamente, se necesitan soluciones sencillas que sean asimiladas por todos. Sin embargo, para mantenerse en posición, se puede considerar la siguiente regla inicial.

A lo largo de una pendiente que comienza con equipo y termina con la funcionalidad comercial, el uso de la innovación tecnológica propietaria se deberá restringir a un lado de la moneda.

El énfasis en las normas y la portabilidad deberá colocarse del otro lado.

En el nivel de equipo/programas del sistema, elementos como Interfaces Gráficas del Usuario (Graphical User Interfaces), Interfaces de DBMS (Database Management System; Sistema de Administración de Base de Datos) e interfaces del Sistema Operativo, permiten realizar mejoras a cambio de explotar las interfaces propietarias. En un segundo nivel, encontramos el porcentaje más alto de la inversión de aplicación y por lo tanto éste deberá recibir la mayor protección de cambios costosos y destructivos.

Por lo tanto, existen varias definiciones laborales de tecnología de sistemas abiertos, algunas promueven la evolución y la innovación y otras ayudan a proteger las inversiones actuales o propuestas en soluciones de la tecnología de información. Muchas metas creadas por estas definiciones se pueden lograr con las normas de los integradores de sistemas que pueden tardar en ser aceptadas y expresadas por completo aunque también formen parte de la solución.

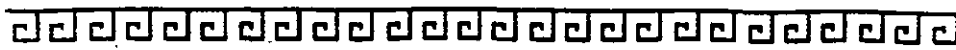




UNIX: BREVIARIO HISTORICO

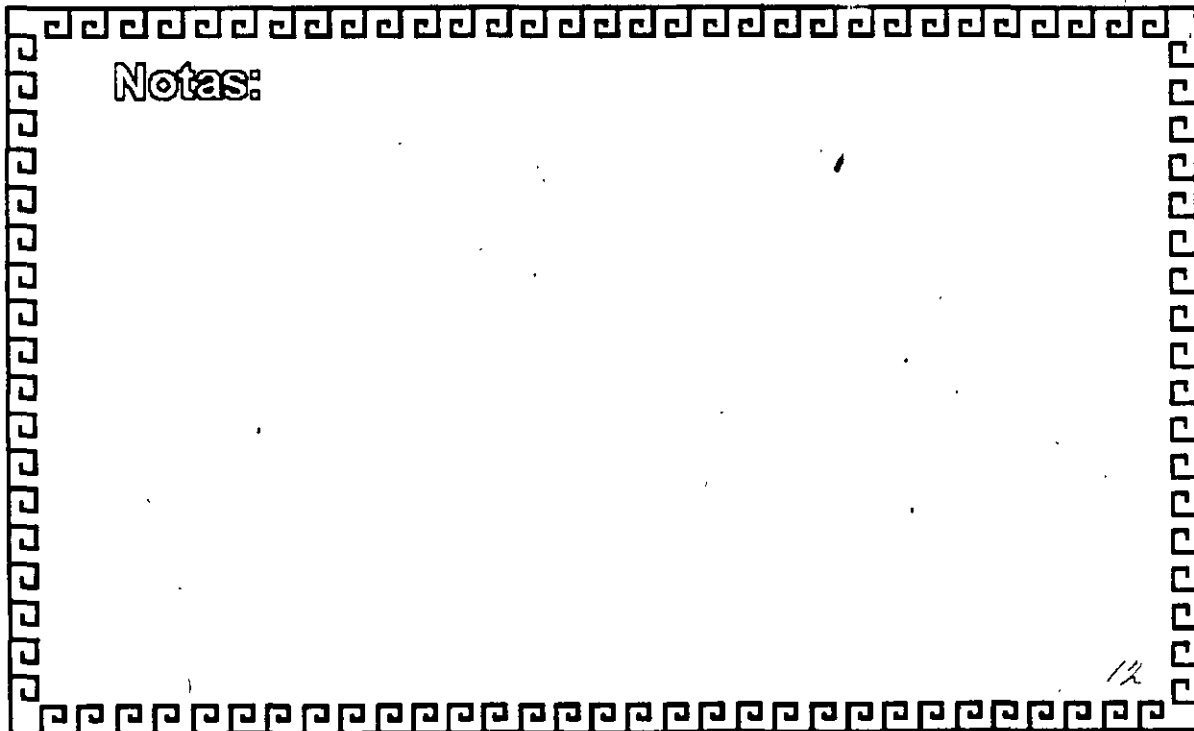
- * Originado en los laboratorios BELL AT&T, antecesor Sistema Operativo MULTICS finales de los 60's
- * KEN THOMPSON y DENNIS RITCHE diseñadores originales constituyen un juego de viaje espacial para la PDP-7
- * Posteriormente crearon una nueva estructura de sistemas de archivos añadiendo entorno de procesos con planificación

Notas:



UNIX: BREVIARIO 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 se desarrolla el lenguaje de programación "C"
- * 1971 primer cliente real fue la Oficina de Abogados de patente BELL con el programa "Troff"
- * En 1973 el KERNELL fue recodificado en "C"



Notas:

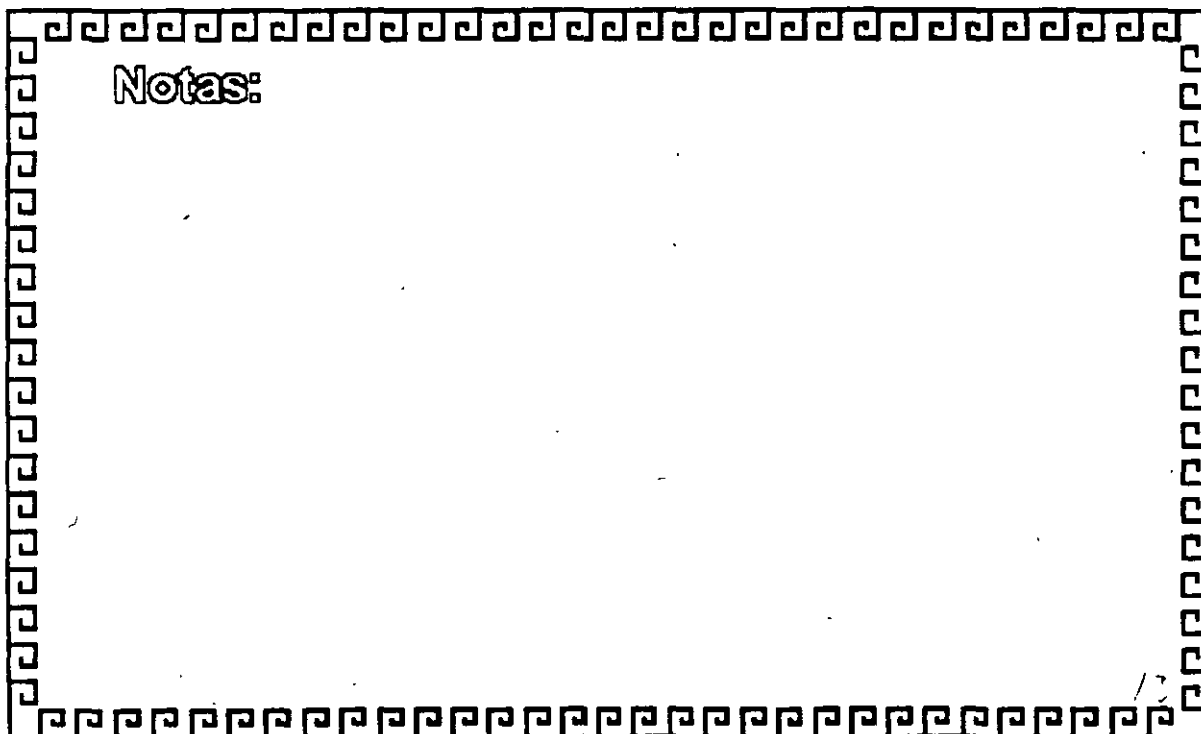


UNIX: BREVIARIO HISTORICO

- * 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 Distribution) en la Universidad de California de Berkeley



Notas:



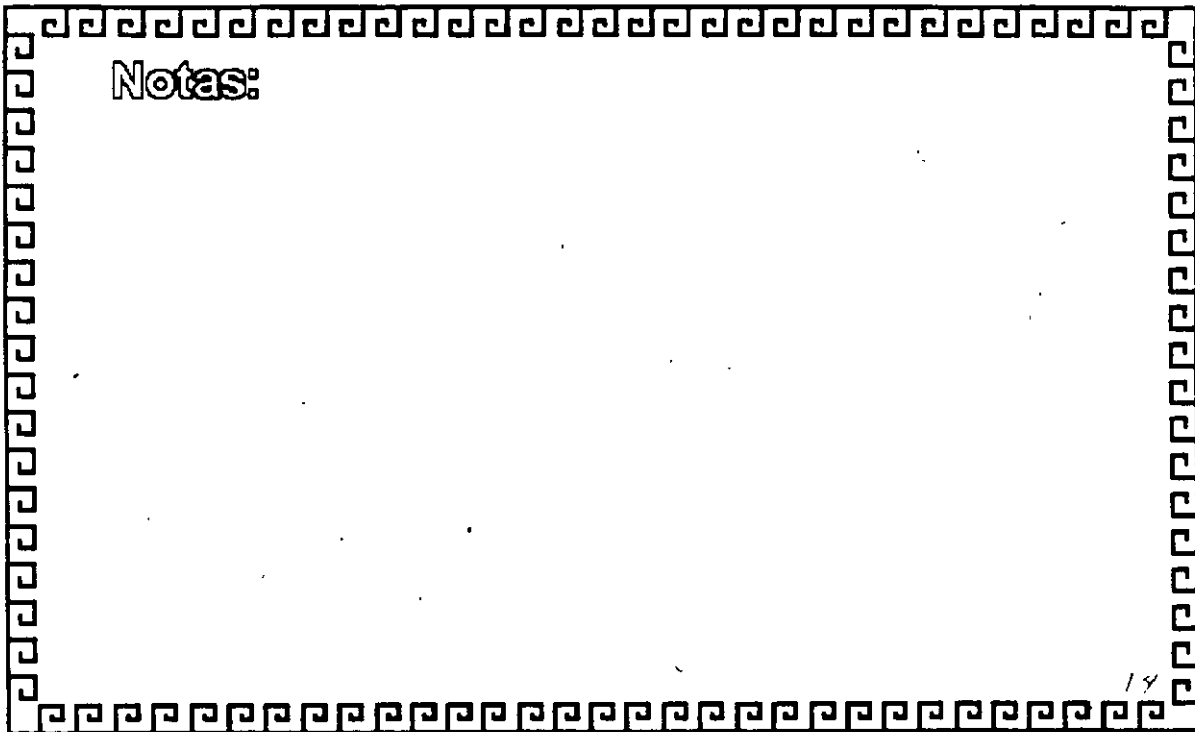


UNIX: BREVIARIO HISTORICO

- * 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
sólo productos de transición

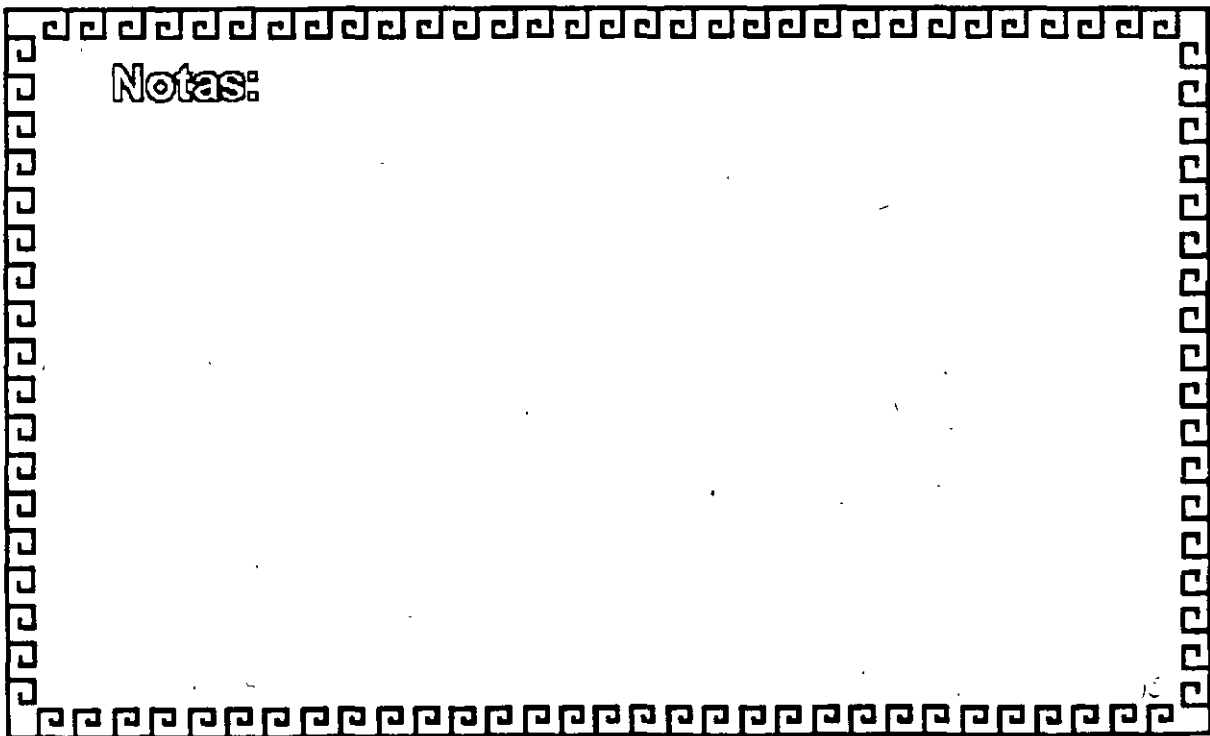
Notas:





UNIX: BREVIARIO HISTORICO

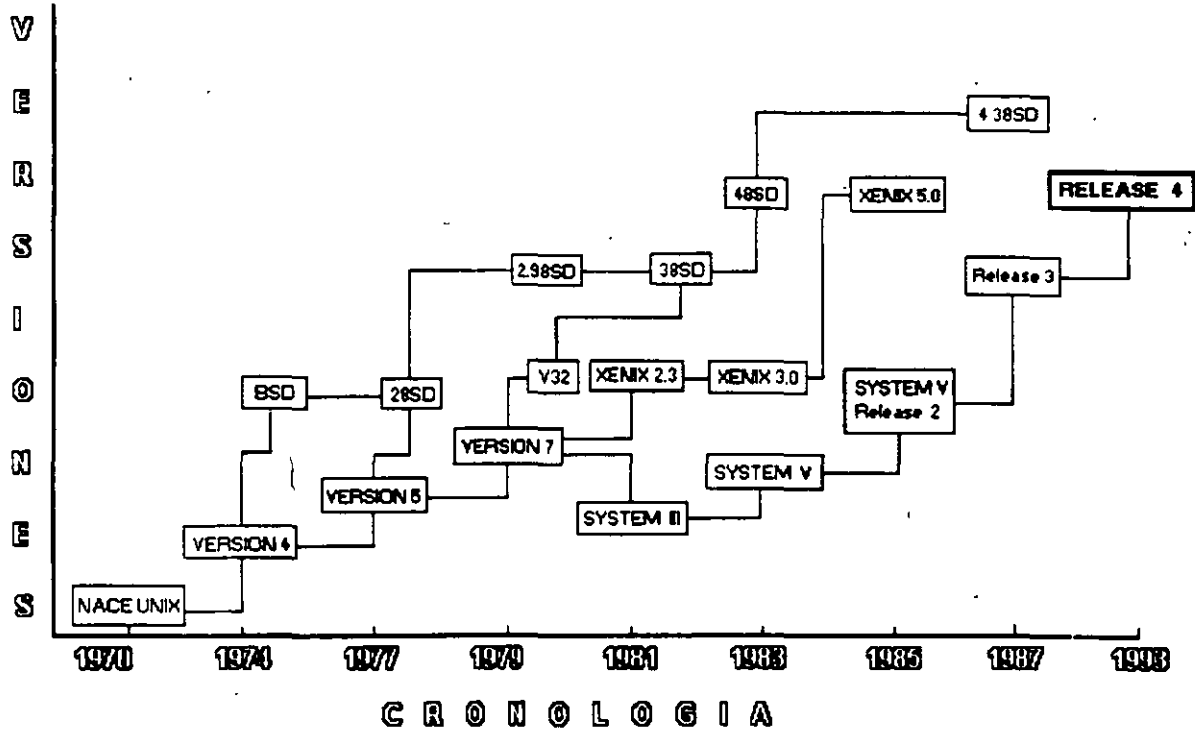
- * Finales de los 70's y principios de los 80's UNIX fue portado prácticamente a casi todas las máquinas con potencial para soportarlo
- * 1986 se genera XENIX para equipos basados en el 8088 con participación de MICROSOFT
- * 1989 Santa Cruz Operation libera su versión SCO UNIXSystem V
- * 1993 Univel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



Notas:



EVOLUCION DE UNIX



Notas:

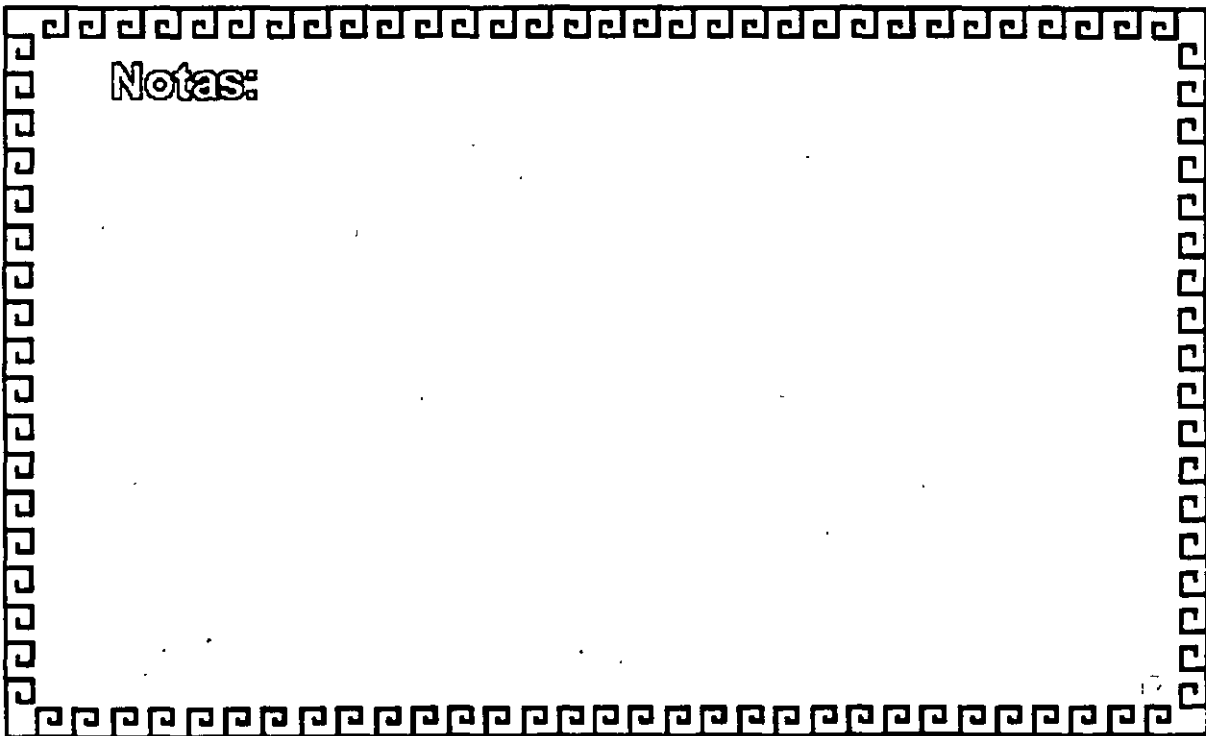


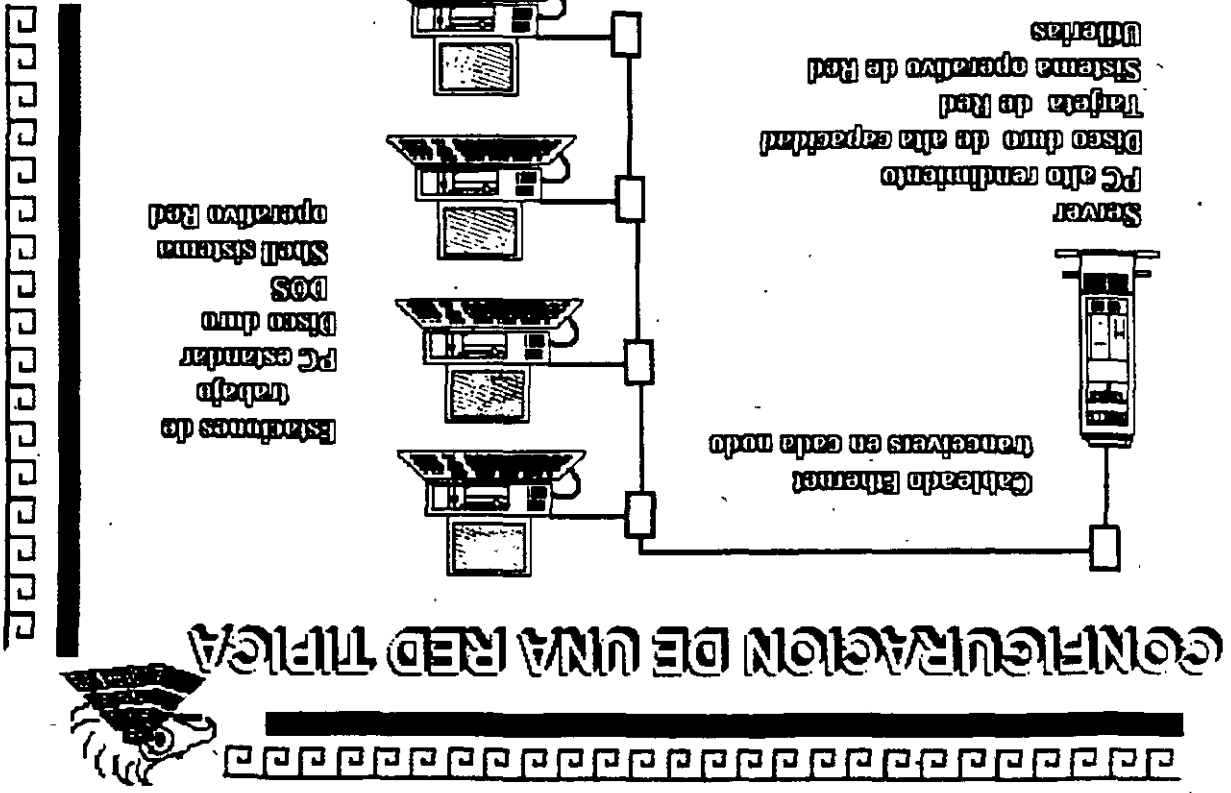
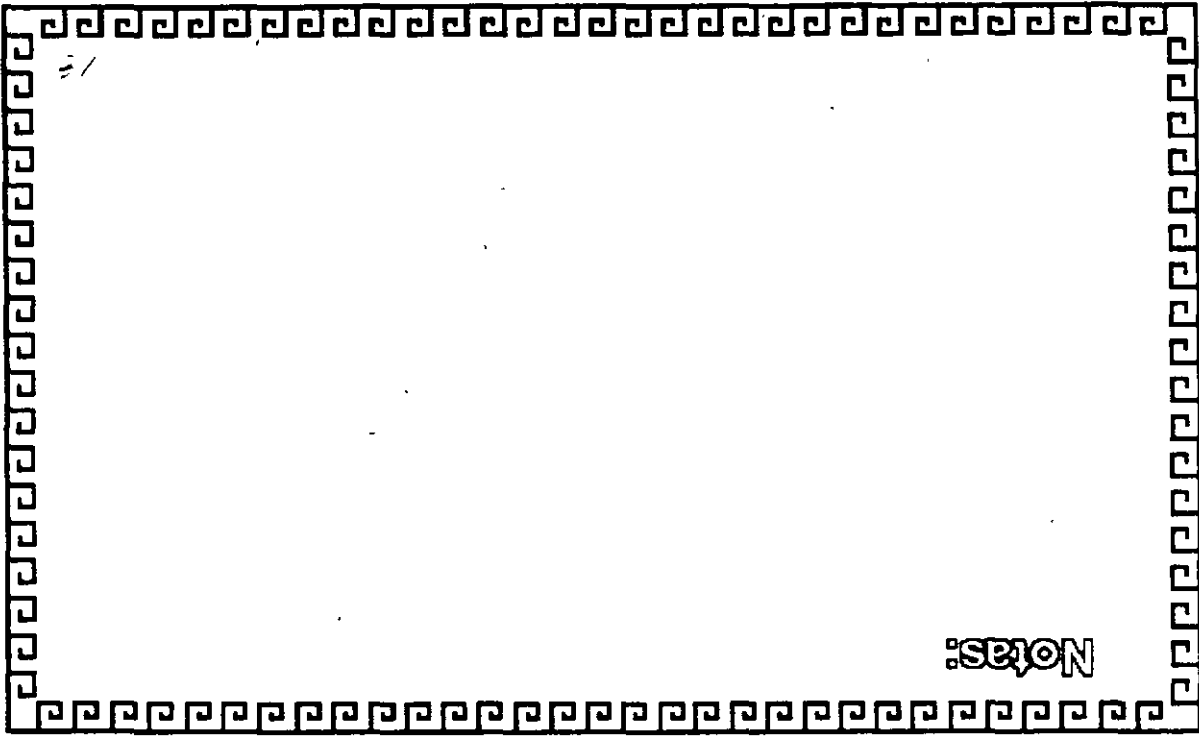
UNIX: "PROTAGONISTAS" IMPORTANTES

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

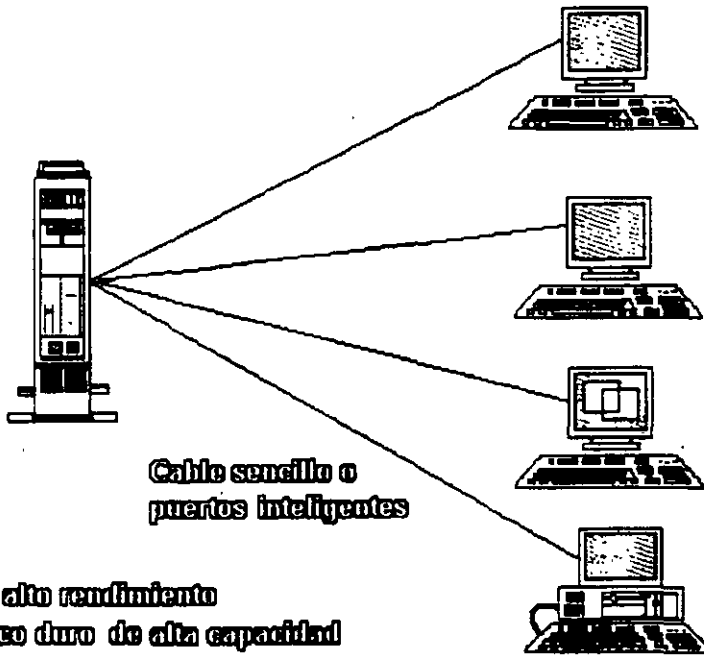
* HP, IBM, USL, DEC, Apollo, Siemens, Nixdorf, Grupo Bull

Notas:





CONFIGURACION TIPICA DE UNIX

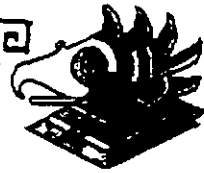
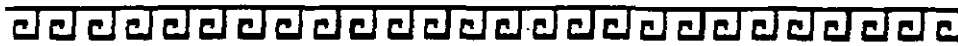


Cable sencillo o
puertos inteligentes

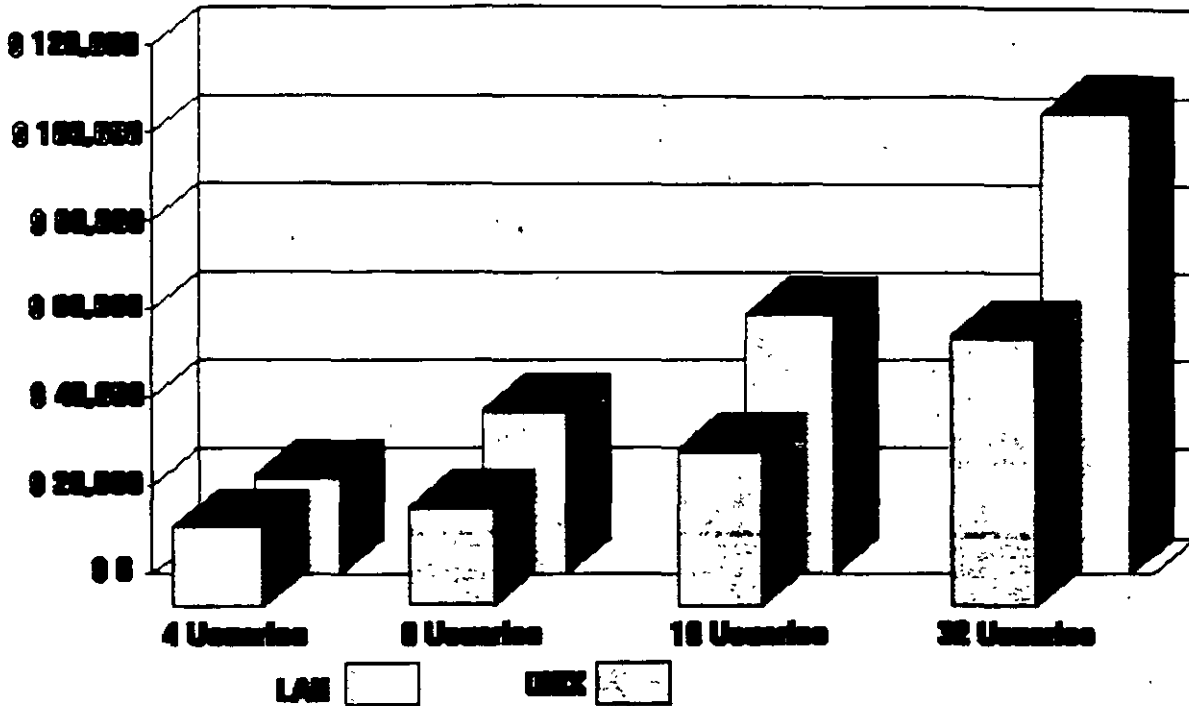
PC alto rendimiento
Disco duro de alta capacidad
Tarjeta multipuertos
Sistema operativo UNIX

Terminales

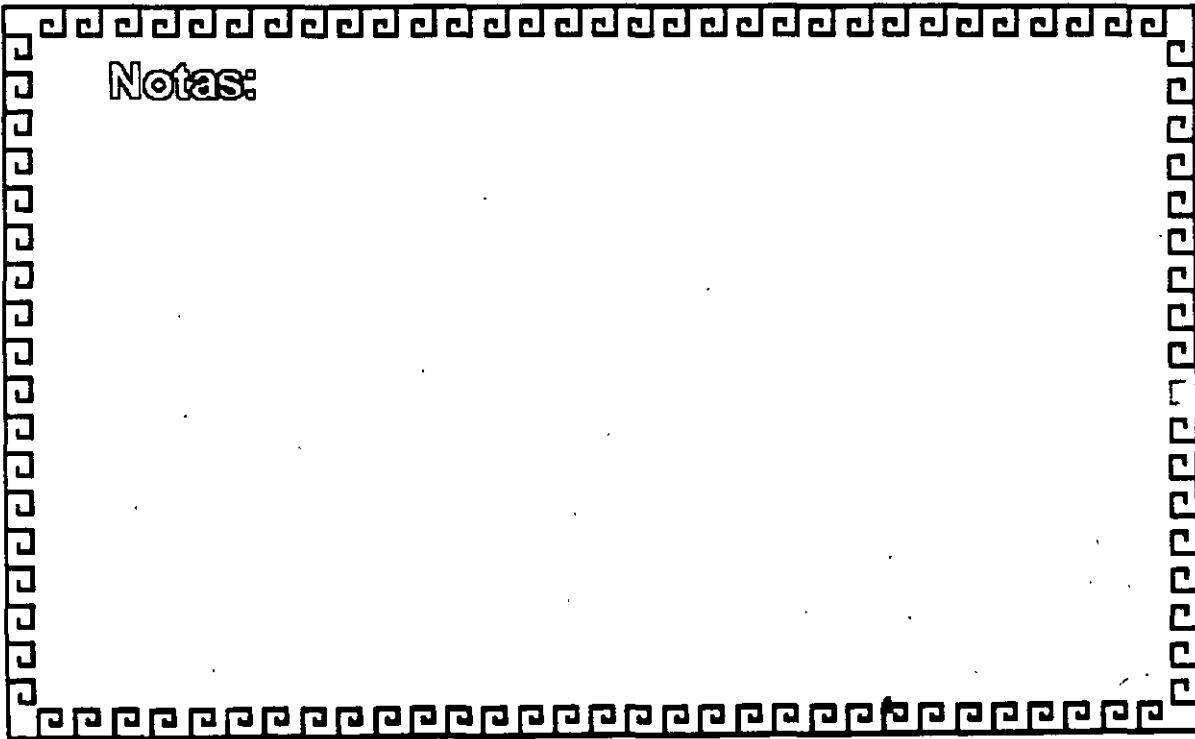
Notas:



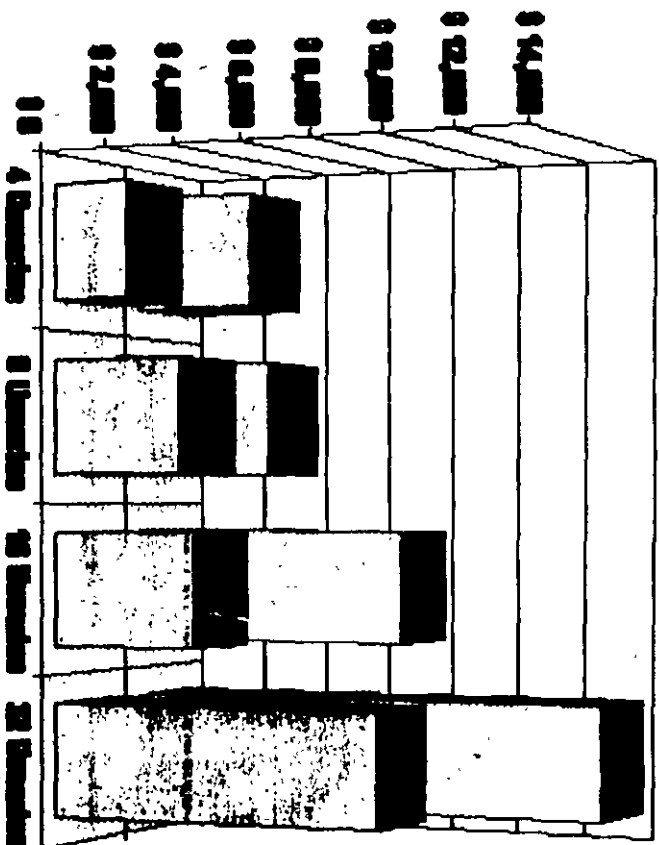
COSTOS POR SISTEMA



Notas:



COSTO ANUAL MANTENIMIENTO Y SOPORTE



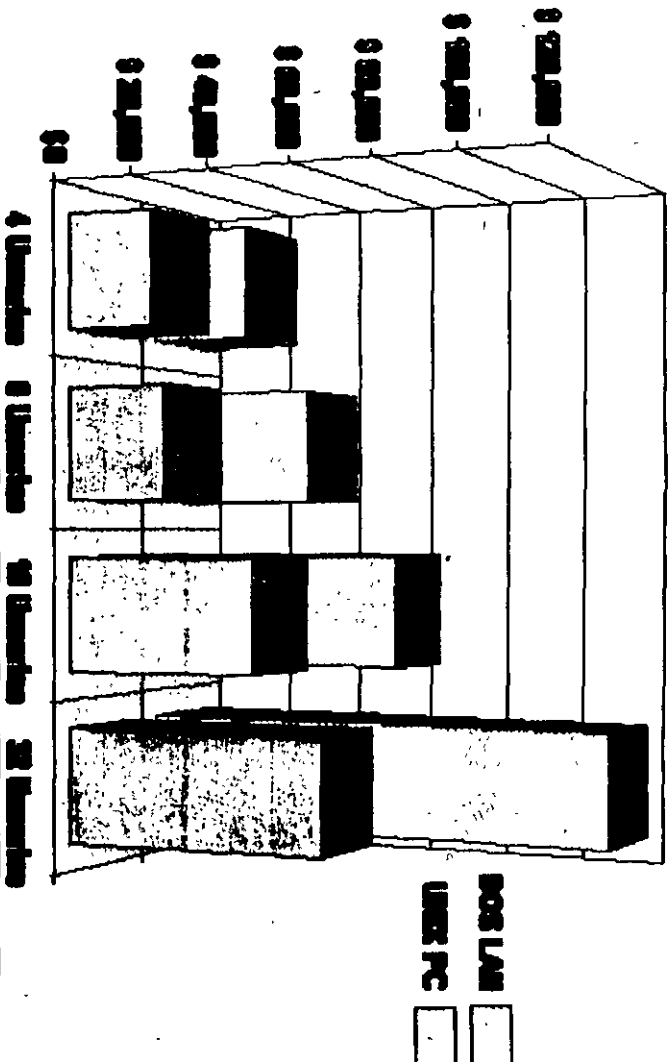
LMI
 MEX

Notas:

Empty space for notes, enclosed in a decorative border.

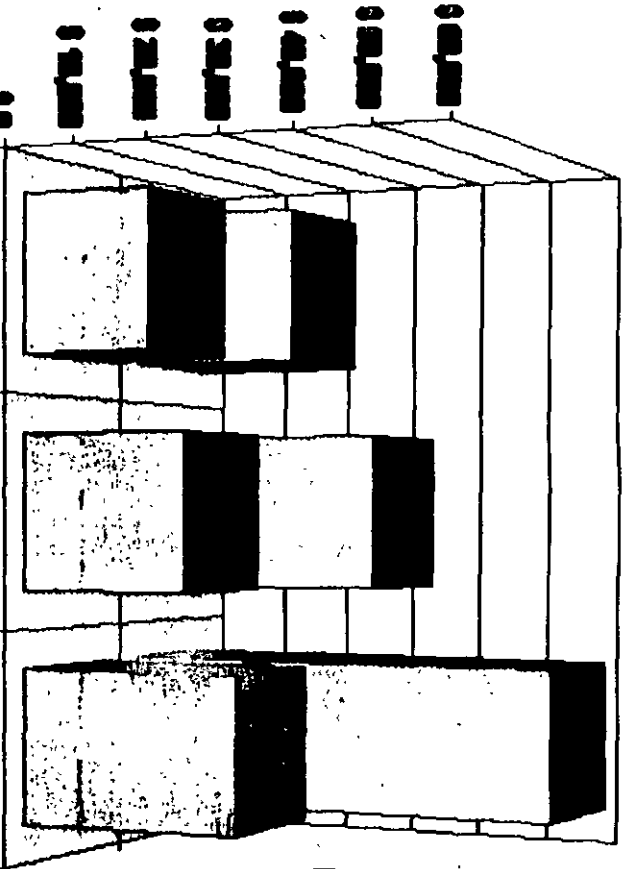


COSTOS DEL SISTEMA DEL PRIMER AÑO



Notas:

ACTUALIZACIONES



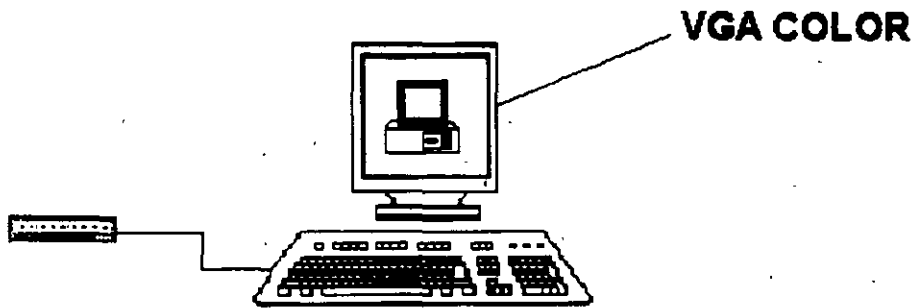
Notas:

[Empty space for notes]



CONCEPTOS GENERALES

TERMINALES X

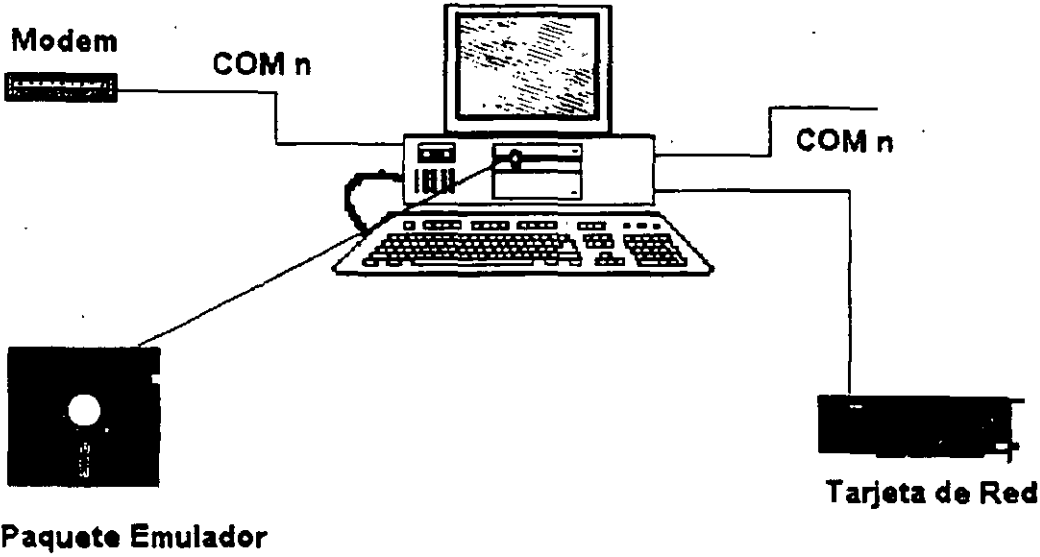


Notas:



CONCEPTOS GENERALES

PC EMULANDO TERMINAL

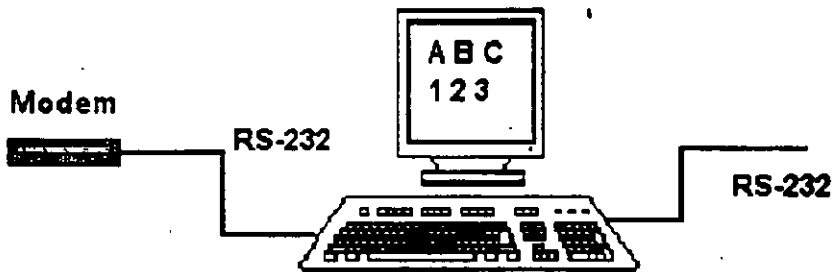


Notas:

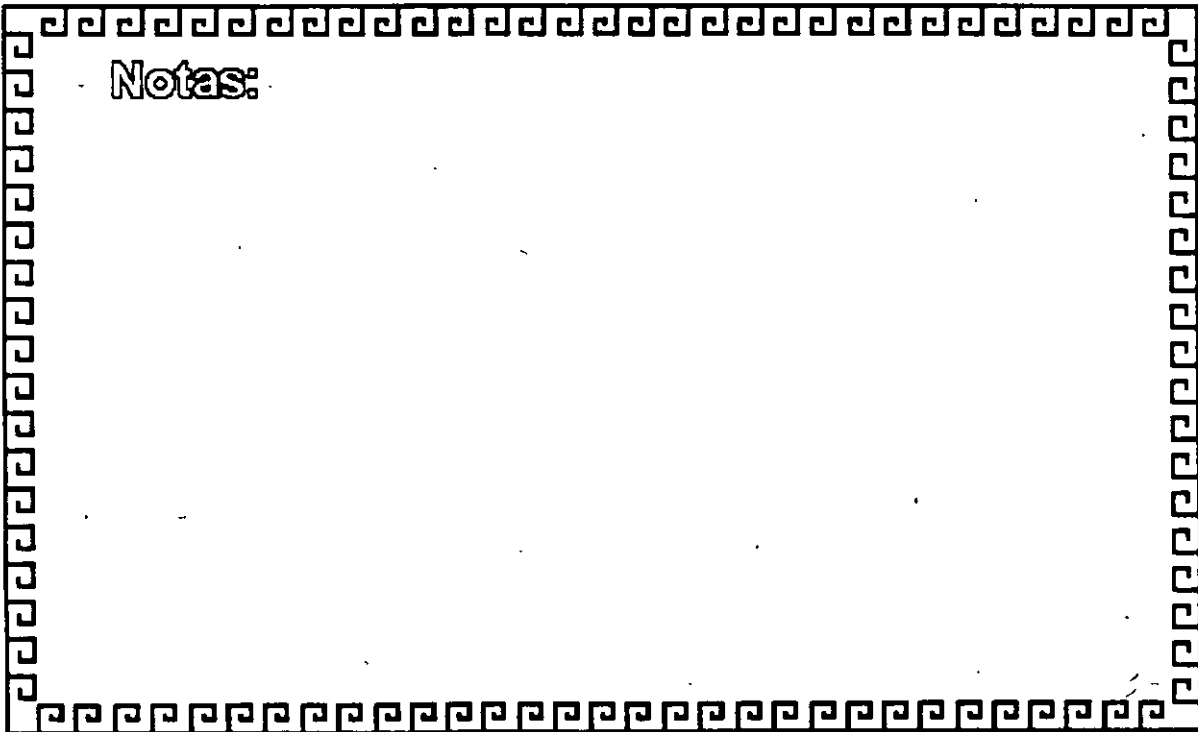


CONCEPTOS GENERALES

TERMINALES TEXTO



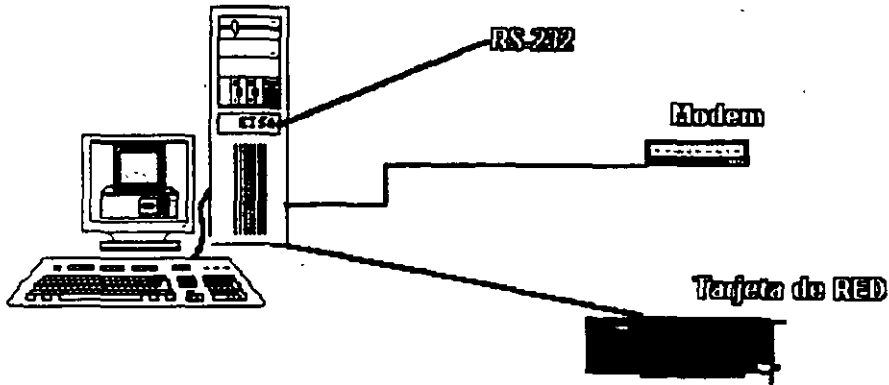
Notas:



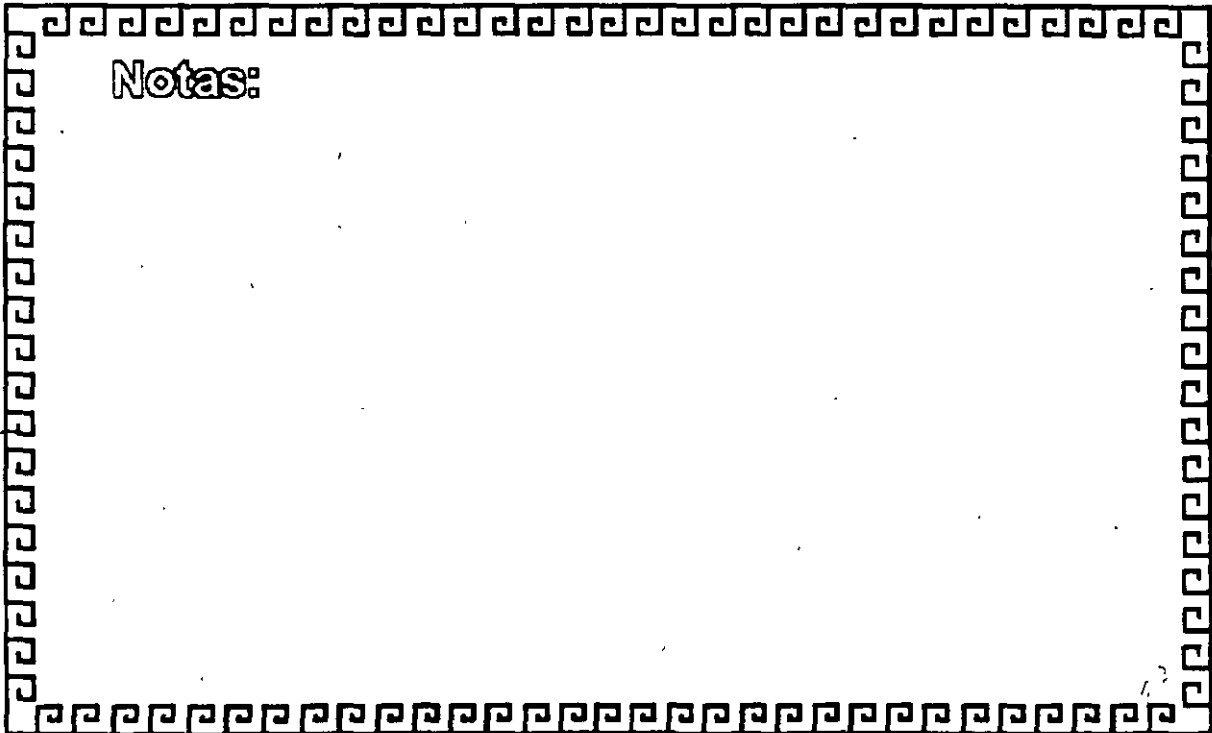


CONCEPTOS GENERALES

PC EMULANDO X-TERMINAL



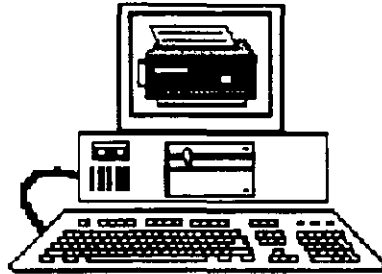
Notas:





' WORKSTATIONS '

Estaciones de trabajo



Notas:

26

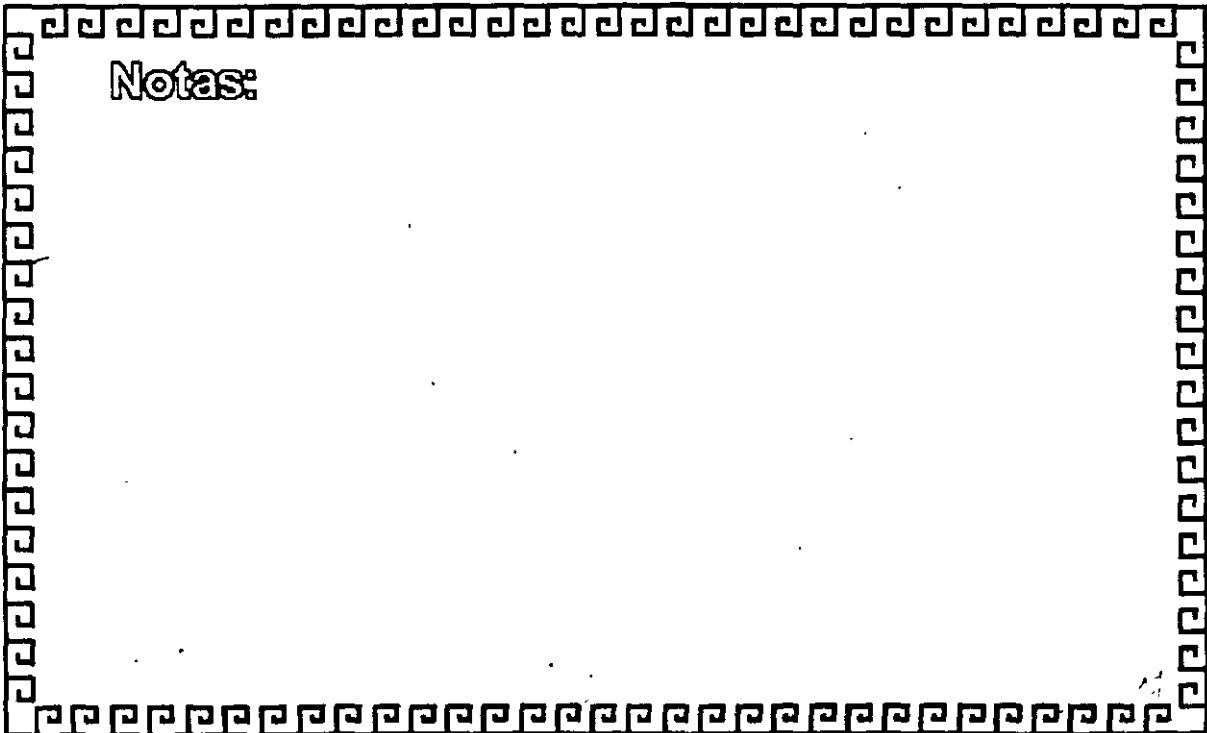


SISTEMAS ABIERTOS

- * No existe una definición absoluta
- * Corre bajo UNIX
- * Se adecuan a las normas internacionales
- * Tienden a evolucionar
- * Son capaces de integrarse



Notas:



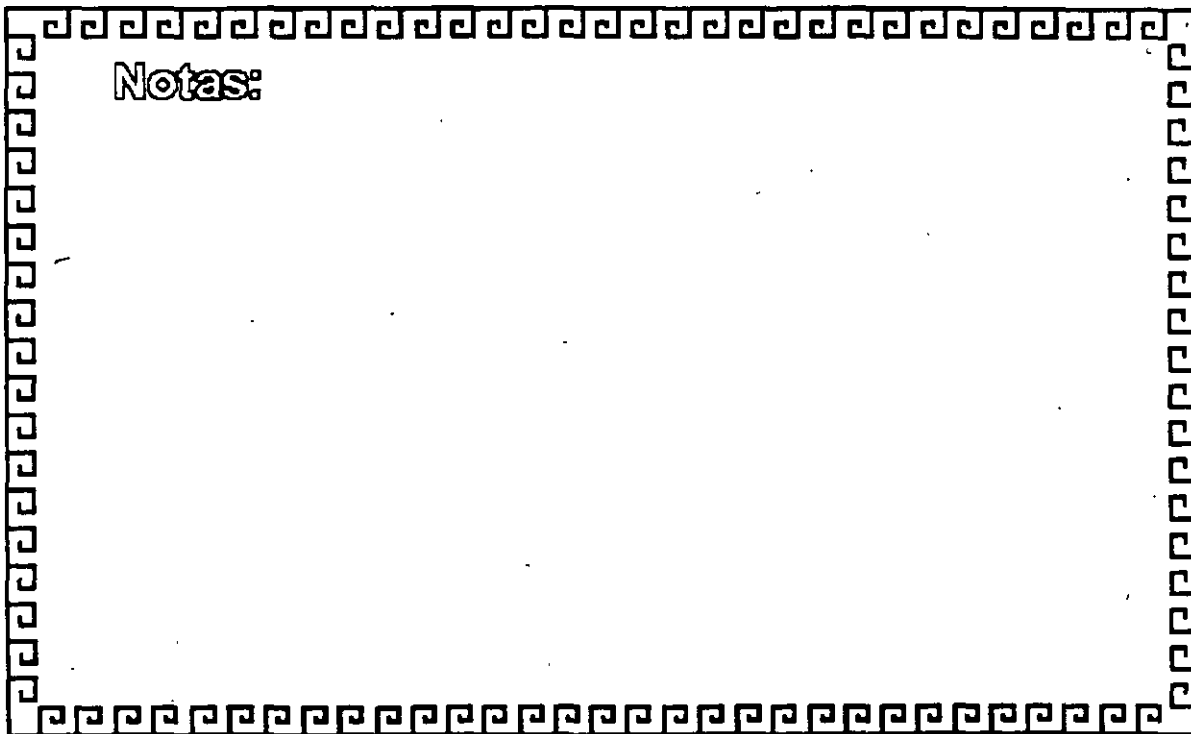


SISTEMAS ABIERTOS

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



Notas:





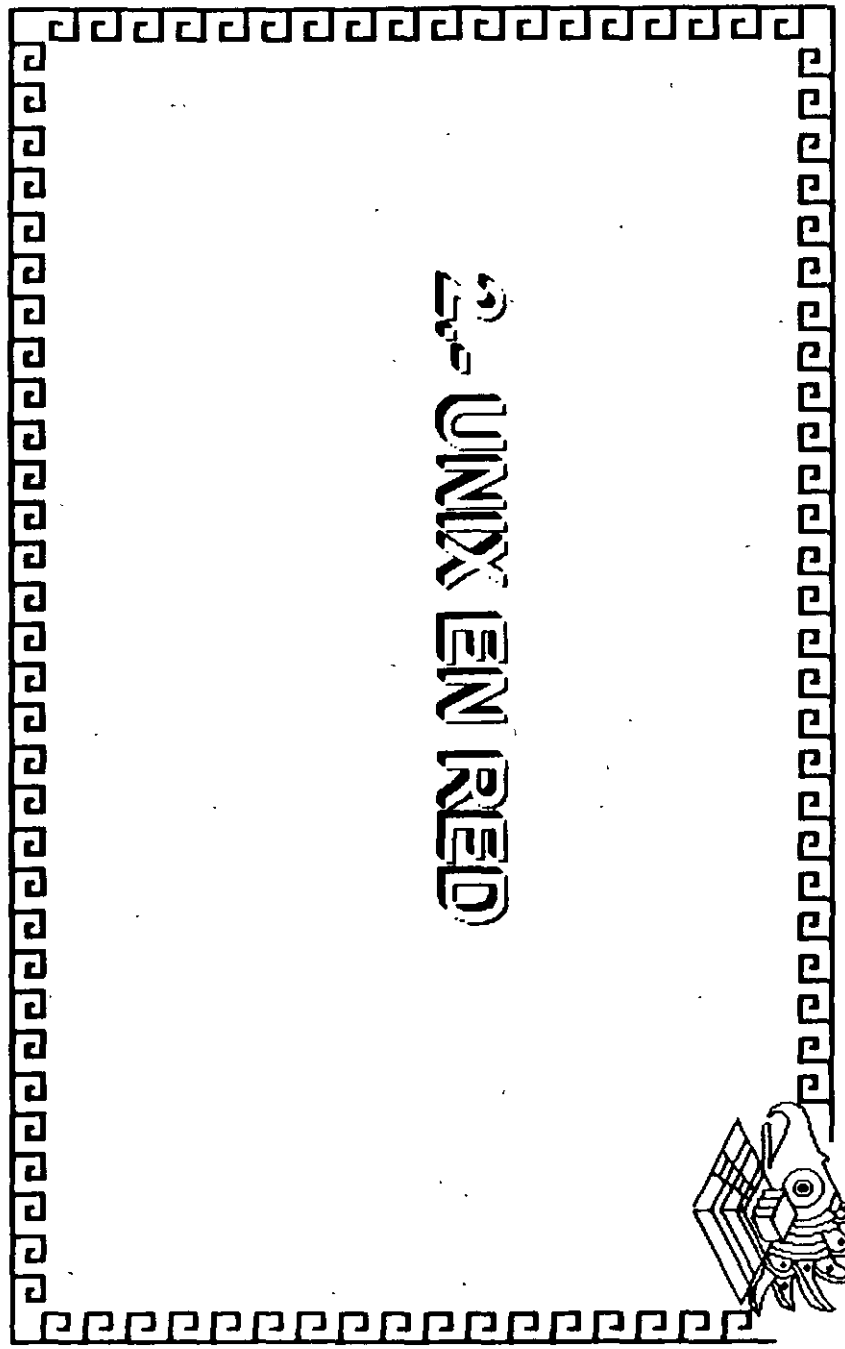
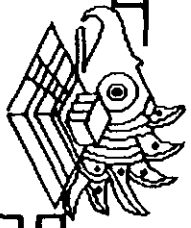
INSTALACIÓN Y MANEJO DE REDES (LAN) EN PLATAFORMA UNIX

2.- UNIX EN RED



Mayo de 1996.

2 UNIX EN REED



UNIX EN RED

CARACTERISTICAS EN AMBIENTE DE RED.

El tipo de red bajo Unix que empieza a aparecer, consiste en enlaces principalmente Ethernet o Token Ring, con equipos tipo cliente basados principalmente en PCs compatibles y equipos servidores que son "workstations": o multiusuarios basados en Unix.

La razón de utilizar PCs como cliente y no algún otro equipo, consiste en la necesidad de muchos usuarios de poder correr las aplicaciones tradicionales de DOS en las PCs. El ambiente de las PCs entonces debería consistir en un sistema de ventanas tipo Microsoft que puede lanzar aplicaciones tanto de DOS como de Unix. Las PCs también requieren el servicio de archivos e impresoras; es decir, el usuario de la PC debería acceder 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 capacidad de lanzar aplicaciones de DOS, Unix caracter y Unix gráfico (X11) desde cualquier servidor en la red y verlas en distintas ventanas de la PC, la PC se vuelve un cliente universal.

La razón de utilizar servidores Unix dentro de las redes locales está principalmente en la posibilidad de correr aplicaciones de base de datos en estos servidores conjuntamente con las demás aplicaciones desarrolladas para Unix y Ventanas X11. El servidor Unix puede adicionalmente correr aplicaciones muy robustas. Los actuales proveedores de bases de datos SQL Oracle, Informix, SyB, Ingres y Progress, cuentan con versiones de sus productos para todas las plataformas Unix. Con estas herramientas, los usuarios pueden desarrollar con rapidez sus propias aplicaciones y correrlas en los equipos servidores de la red.

Los servicios que proporcionan los servidores Unix, entonces, son los siguientes:

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



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

- * Poder correr aplicaciones de DOS simultáneamente con aplicaciones gráficas X11 y aplicaciones Unix de carácter.
- * Conectividad. A una Red Ethernet (Lan O Wan) con TCP/IP se pueden conectar:
 - Terminales tontas
 - PCs
 - Terminales X
 - "workstations"
 - Mainframes
- * Heterogeneidad de marcas. Se pueden mezclar marcas diversas con SUN/HP/IBM/DEC.etc.
- * Simetría. Los clientes pueden lanzar aplicaciones de cualquier servidor. Cualquier servidor puede ser también cliente.
- * 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 incorporarlas, ya que se convertirán en tecnologías de punta en los siguientes años.



Estaciones de Trabajo "WorkStations"

Durante los últimos años, se ha visto un crecimiento muy fuerte en el uso de redes locales basadas en servidores Unix. Esta tendencia empezó con la introducción al mercado de las poderosas "workstations" (Estaciones de Trabajo) basadas en la tecnología RISC (computadoras con un conjunto reducido de instrucciones). Los atributos de estas "workstations" hacen muy atractivo su uso como servidores en las redes locales, además de su tradicional orientación a aplicaciones de gráficas (CAD/CAM), desktop publishing, CASE y diseño.

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

1.- PROCESADOR PODEROSO DE 32 BITS BASADO EN TECNOLOGIA 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 mejorar muchos procesos simultáneos en modo multiusuario.

2.- PANTALLA GRAFICA GRANDE Y RATON (MOUSE)

Todos los modelos "workstations" cuentan con pantallas gráficas de 19" y generalmente de color. Las imágenes manejadas son "bit-mapped", es decir que lo que se ve en la pantalla es un reflejo de un arreglo 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.

3.- TARJETAS ETHERNET O TOKEN RING INTEGRADAS

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



4.- SISTEMA OPERATIVO UNIX CON VENTANAS X11

Las distintas marcas de "workstations" en el mercado tienen otro atributo que les da cierta compatibilidad: todas cuentan con el sistema operativo Unix, y el sistema de ventanas X11. Unix que originalmente se desarrolló en los Laboratorios Bell de AT&T, es un sistema operativo multitarea y multiusuario. Es robusto y se ha vuelto el estándar para equipos multiusuario de tamaño mediano. A través del sistema de ventanas X11, diferentes modelos de "workstations" pueden coexistir en la misma red local y compartir aplicaciones mutuamente. Con otro producto, NFS (Network File System; Sistemas 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 permite ver a una red local como un sólo 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 Massachusetts (M.I.T.) a partir de 1985 y proviene de un 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 e IBM. Actualmente se encuentra en la versión 11.5.

El paquete X11 consiste en una serie de subrutinas para el manejo y despliegue de imágenes con funciones para crearlas, expandirlas, moverlas, etc., y además 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 coexistir 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 su propia pantalla. En este caso, la aplicación corre en la otra computadora mientras el servidor X está 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 prom. Al conectarse en la red, estos dispositivos, llamados Terminales X, pueden correr aplicaciones en otras "workstations" en la red y verlas en su pantalla. Existen programas también que se pueden instalar en PCs, convirtiéndolas en terminales X.



5.- GRUPOS DE TRABAJO

Esta posibilidad, de tener una aplicación corriendo en un equipo y el servidor X en otro, ha creado un nuevo concepto en la computación moderna, el de un "grupo de trabajo". En este concepto, varias "workstations" 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 sólo resultado final.

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

FABRICANTES DE "WORKSTATIONS"

Los principales fabricantes de "workstations" son Sun Microsystems, Hewlett Packard, Digital Equipment e IBM.

SUN MICROSYSTEMS

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

HEWLETT PACKARD

HP adquirió a la empresa APOLLO, otro fabricante de "workstations" 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 uno 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. En 1991, alcanzó el 20% del mercado.



DIGITAL EQUIPMENT

DEC cuenta con una línea de "workstations" llamada Decstations, basada en el chip procesador RISC de la empresa MIPS. Se formó una alianza de más de 30 empresas denominadas ACE, (Advance Computer Environment) para fabricar clones usando la nueva versión de este chip MIPS4000. Actualmente DEC liberó un chip "Alpha" de 64 bits con posibilidades de superar a los 200 MIPS. A raíz de esto, DEC probablemente dejará el consorcio ACE.

IBM

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

OTROS

Existen otros fabricantes de "workstations" como Sequent Silicon, Graphics, CDC, etc/ cuya fracción del mercado es de 15%.

Es importante recalcar, que en la actualidad existe una verdadera guerra de precios entre todos estos fabricantes, con las consecuencias lógicas: baja de precios, aumento de la tecnología y poder de cómputo.

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 "Xoftware for DOS" que hace esta función. Al utilizar el ambiente Windows de Microsoft JSB Multiview Desktop/X, es posible que un usuario de una PC conectada en una red de "workstations" 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 casas de software, integradores de sistemas y usuarios finales.

El equipo personalizado de desarrolladores de Locus trabaja 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 diseñan utilerías para el mismo.

Locus también ofrece al cliente un laboratorio el cual cuenta con distintas plataformas, sistemas operativos para sus pruebas, además ofrece asesoría para una integración completa de su desarrollo.

Actualmente existen más de 500,000 instalaciones de Locus Computing Corporation en todo el mundo de los siguientes productos:

PC-INTERFACE:

Es un software con características de red el cual permite a usuarios con PCs y/o Macintosh compartir servicios como son sistemas de archivos, recursos de impresión desde servidores Unix y/o Xenix.

Los sistemas de archivos son obtenidos desde el servidor; en el caso de que la PC y/o Macintosh no cuenten con disco duro, el PCI podrá asignar 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 Unix y/o Xenix sin importar que la aplicación esté en DOS.

Actualmente está liberada la versión 4.1 de PC-Interface la cual ya contiene drivers (NOIS.DRV) el cual da soporte a Novell.



BENEFICIOS:

- * Requerimientos de memoria mínimos.
- * Servidor Unix y/o Xenix no dedicado.
- * Seguridad completa de información a través de Unix para DOS.
- * Capacidad para manejar múltiples sistemas Unix.
- * Emulación de terminal VT220/VT100 para PCs.
- * Emulación de terminal VT320/VT102 para Macintosh.
- * Ejecución de procesos remotos.
- * Ejecución de comandos Unix desde DOS y/o Mac.
- * Soporta PCs remotas.
- * Soporta MS Windows 3.0.
- * Soporta tarjetas Ethernet, Token Ring y puerto de comunicaciones RS-232.

DRIVERS ETHERNET.

- * 3Com 501,505,523
- * Digital Equipment Corp., DEPCA, DE100, DE200
- * Excelan
- * Racal Interlan
- * Ungerman
- * Western Digital WD8003 E.EB y EBI
- * Western Digital WD8013 EBI
- * Western Digital WD8003 E/A(MCA)
- * Xircom Pocker Ethernet (pandad gemela no es soportada)
- * NW 1000 y NW2000

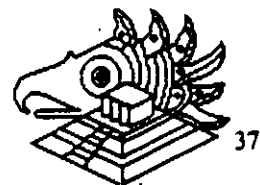
DRIVERS TOKEN RING

- * Tarjetas IBM (4 y 4/16)

SERVIDORES DE PC-INTERFACE INCLUIDOS EN VARIAS MARCAS DE UNIX:

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

Existen 45 distintas plataformas de PC-Interface (servidor) y se cuenta 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 casadas en DOS comunicarse con una gran variedad de servidores Unix y/o Xenix más comunes en la industria, permite establecer sesiones remotas desde la PC transferencia de 4 archivos entre su 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 archivos)
 - TELNET (procesos remotos incluye emulación de terminal VT220 e incluye modos de emulación H19, VF52 y ANZY X.364)
- * Aplicaciones de red distribuidas.
- * Multifunción de estaciones de trabajo.

TCP/IP para DOS soporta usuarios con programas de red utilizando una interface en programas de aplicaciones librerías 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 está soportado el puerto de comunicaciones RS-232.

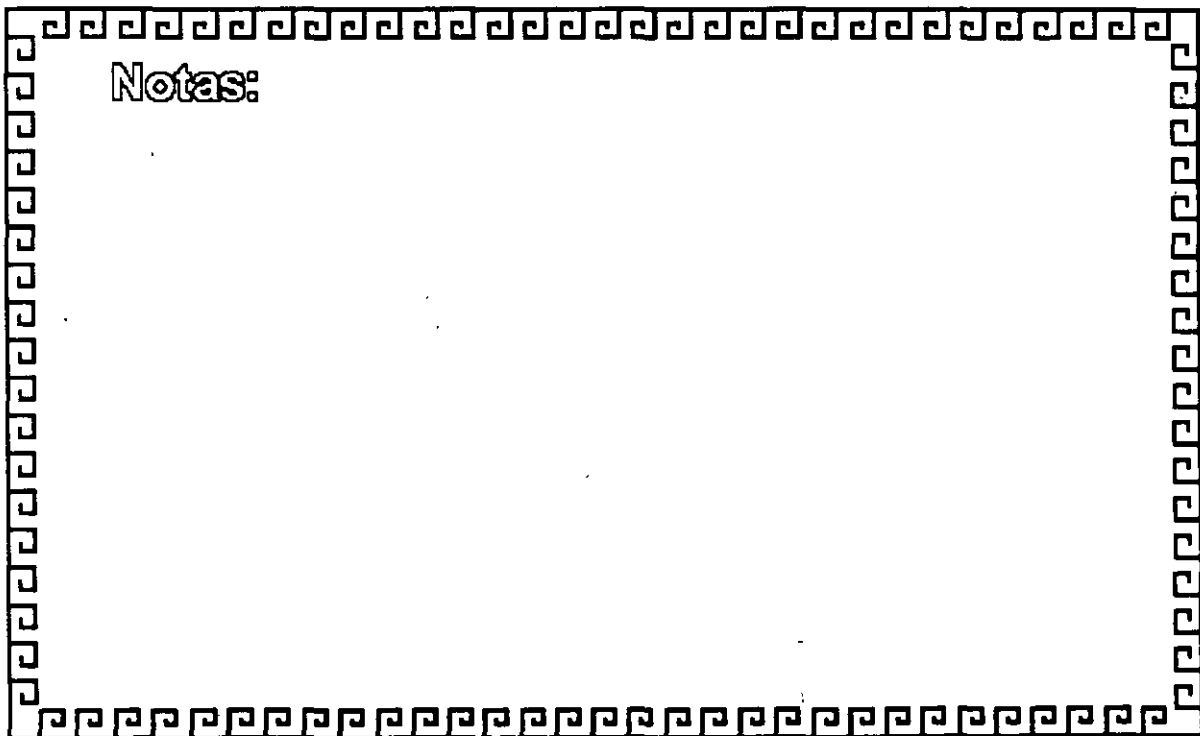


UNIX EN RED

Los servicios que proporcionan los servidores UNIX

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

Notas:



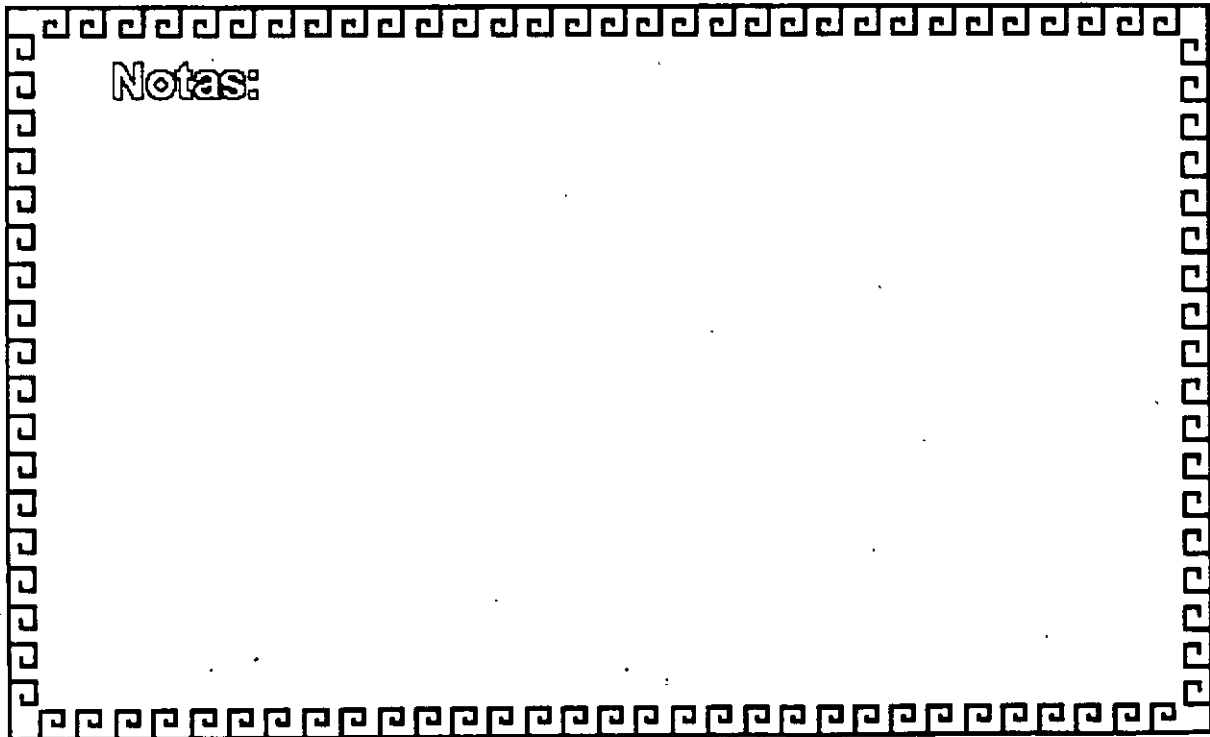


UNIX EN RED

Las ventajas para los usuarios :

- Poder correr aplicaciones de DOS simultáneamente con aplicaciones gráficas X11 y aplicaciones UNIX de caracter.
- Conectividad. A una Red Ethernet (Lan o Wan) con TCP/IP se pueden conectar:
 - Terminales tontas
 - PCs
 - " Workstations "
 - Mainframes
- Heterogeneidad de marcas.
- Simetría. Los clientes pueden lanzar aplicaciones de cualquier servidor. Cualquier servidor puede ser también cliente
- Sistemas abiertos

Notas:





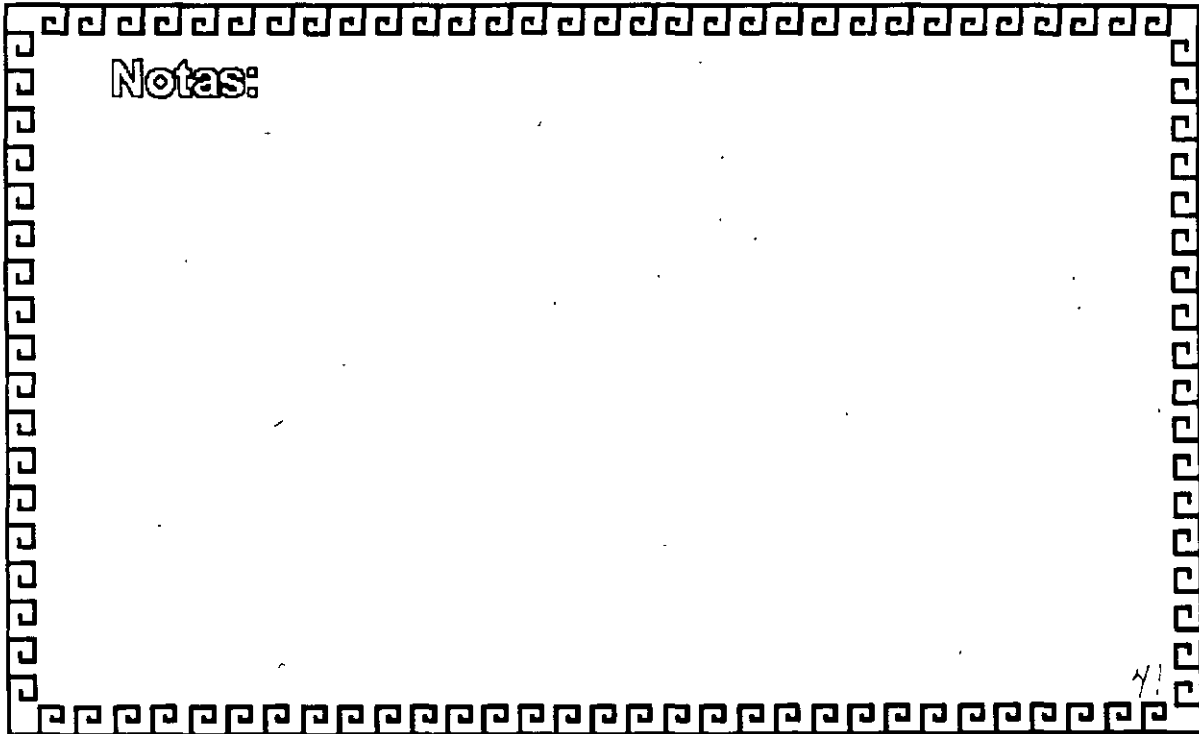
UNIX EN RED

ESTACIONES DE TRABAJO " WORKSTATIONS "

- 1.- Procesador poderoso de 32 bits basado en tecnología RISC
- 2.- Pantalla gráfica grande y ratón (Mouse)
- 3.- Tarjeta Ethernet o Token-Ring integrada
- 4.- Sistema operativo UNIX con ventanas X11
- 5.- Grupos de trabajo



Notas:





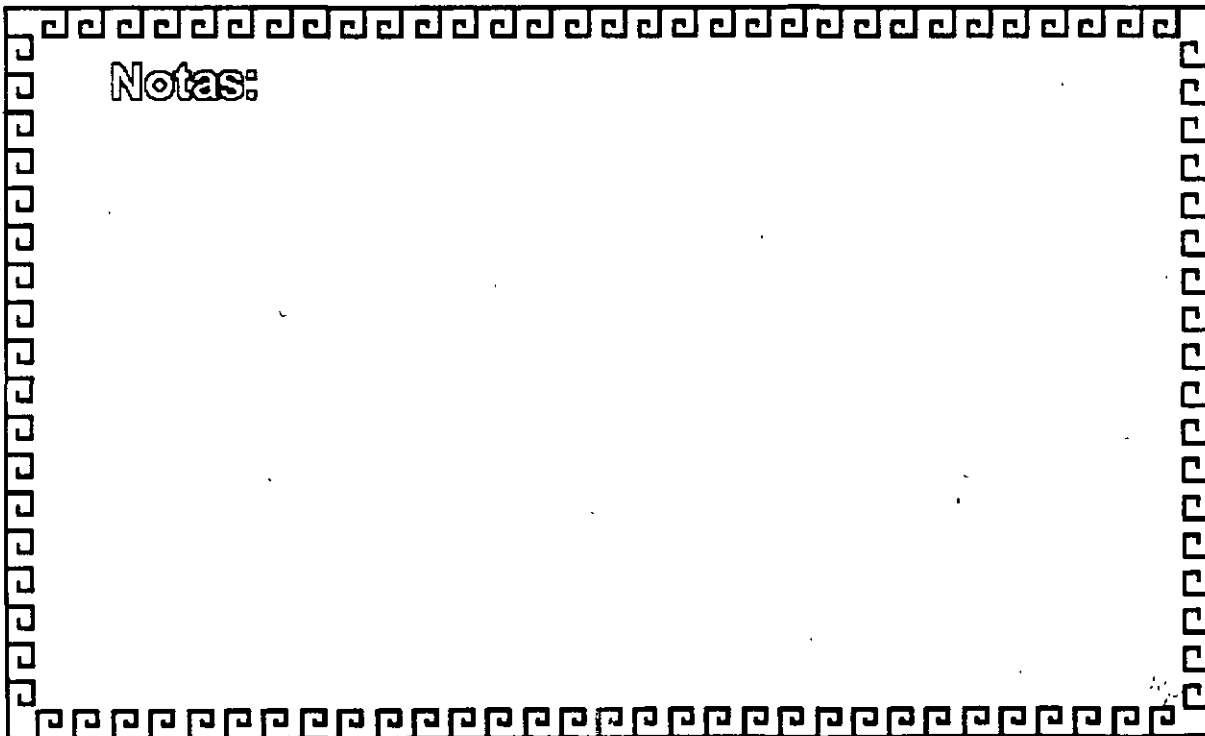
UNIX EN RED

FABRICANTES DE "WORKSTATIONS"

- SUN MICROSYSTEMS
- HEWLET PACKARD
- DIGITAL EQUIPMENT
- IBM
- OTROS



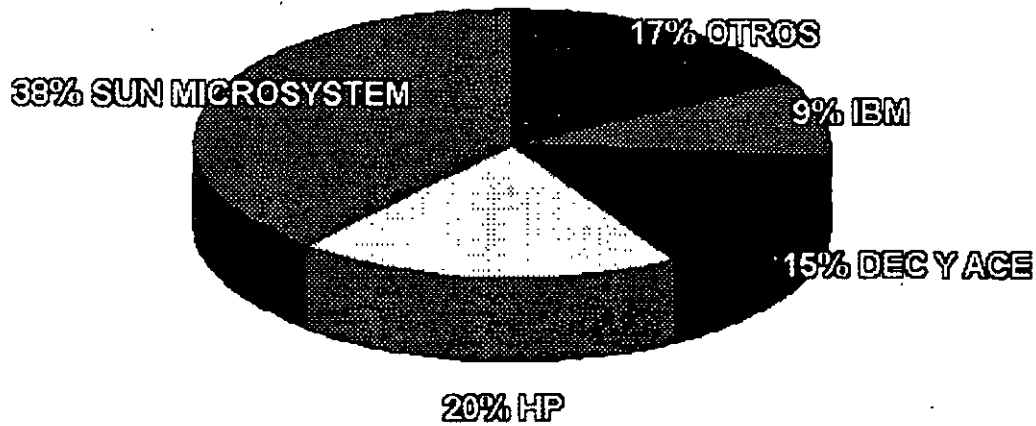
Notas:





UNIX EN RED

FRABRICANTES DE " WORKSTATIONS "



Notas:

43



UNIX EN RED

PRODUCTOS DE SOFTWARE D.O.S PARA CONECTIVIDAD UNIX

Producto:

Software for DOS

JSB Multiview

PC-Interface

TCP/IP para D.O.S

PC XSIGHT

Compañía:

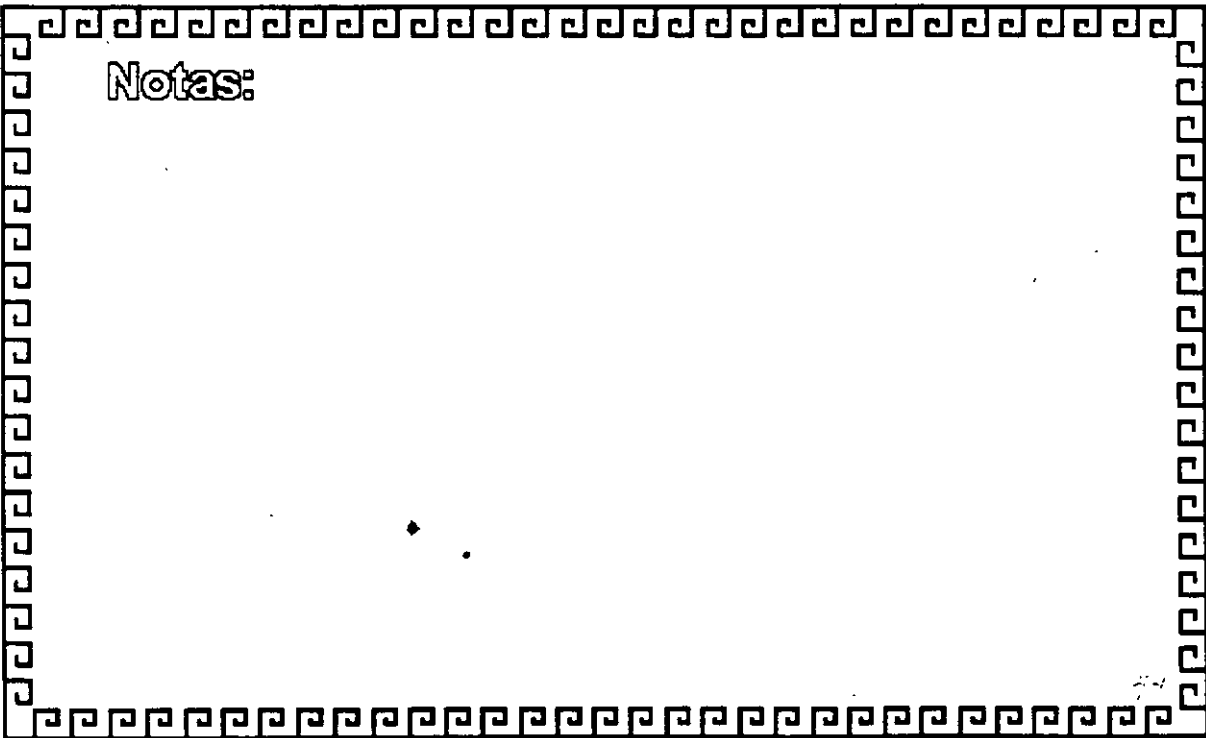
AGE Logic

LOCUS

LOCUS

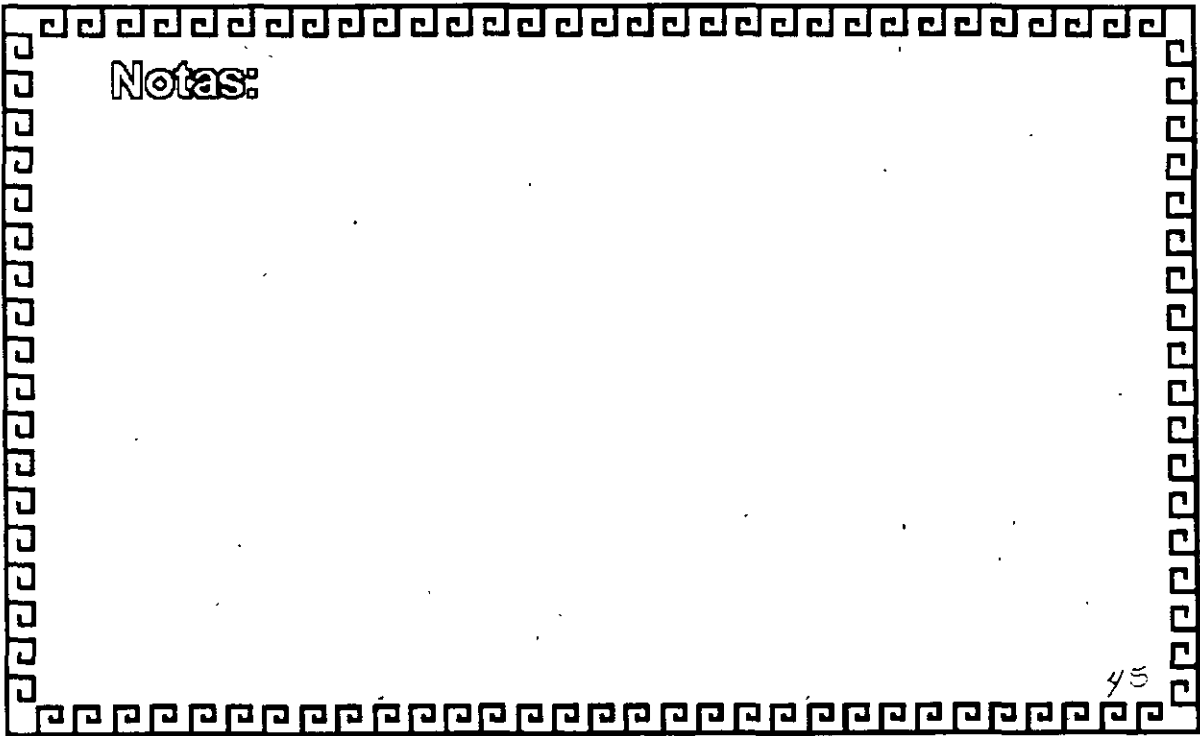


Notas:





HARDWARE DE UNIX EN RED

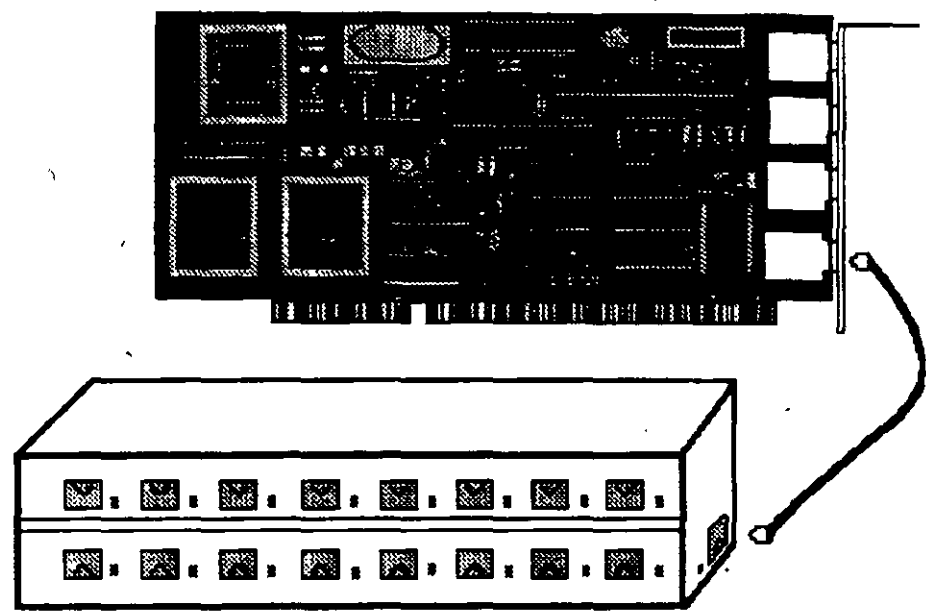


Notas:



UNIX: HARDWARE PARA REDES

TARJETA MULTIPUERTO



Notas:



TARJETA MULTIPUERTOS

PRINCIPALES CARACTERISTICAS

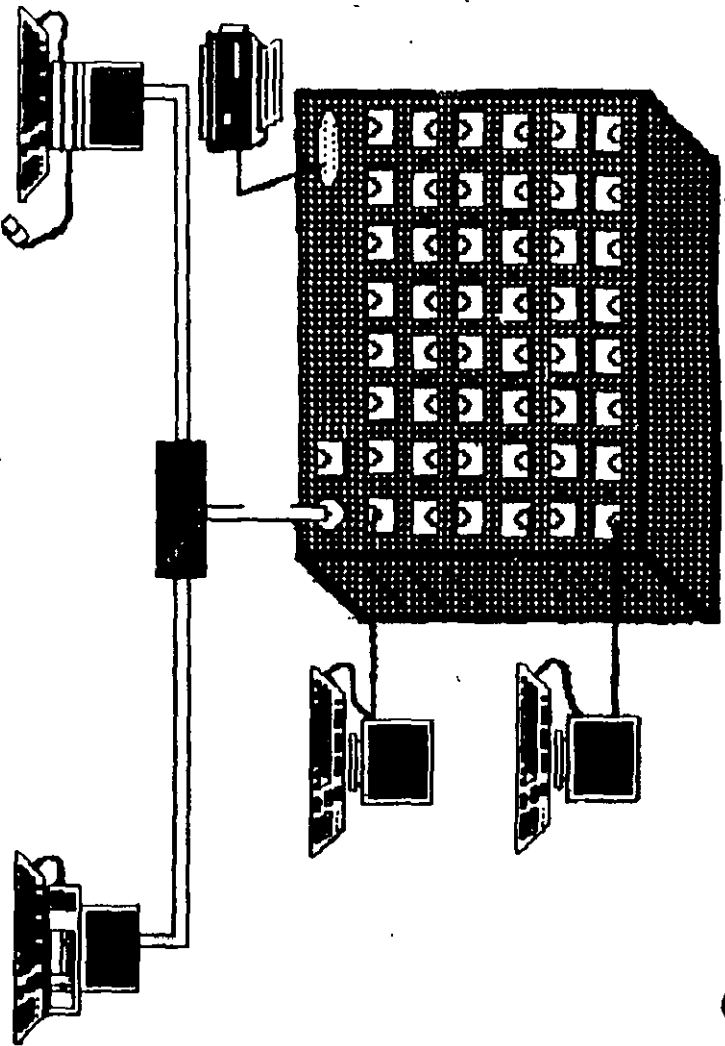
- * No inteligente
- * Inteligente
 - Procesadores: RISC de 10 Mhz a 16 Mhz
 - INTEL 80186 de 10 Mhz a 16 Mhz
- * Memoria: de 64 Kb a 512 Kb
- * Arquitectura: ISA, EISA, MCA, RS6000
- * Número de Puertos: de 4 a 96
- * Velocidad: de 75 bps a 38,400 bps
- * Con o sin soporte a Modem



Notas:

42

SERVIDOR DE TERMINALES



Notas:



SERVIDOR DE TERMINALES

¿QUE 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.



<p>Notas:</p> <p>49</p>



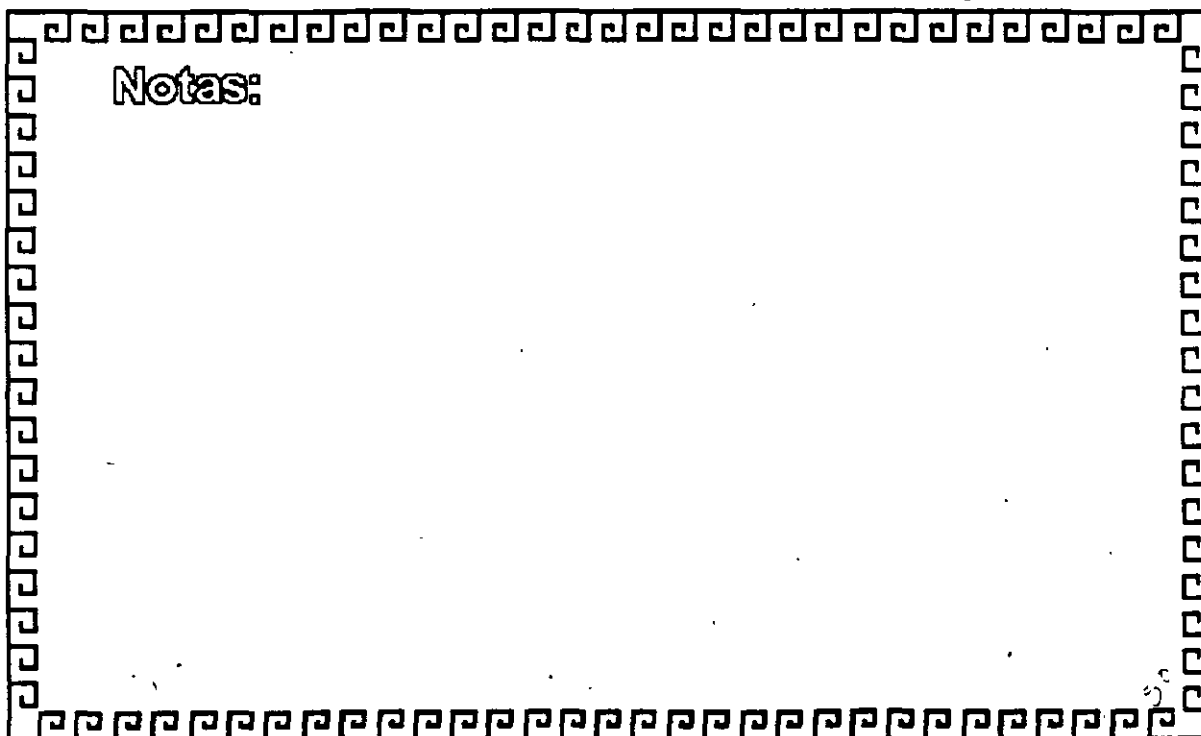
SERVIDOR DE TERMINALES

CARACTERISTICAS PRINCIPALES

- * Procesador: INTEL 80386sx a 16 Mhz
- * Número de puertos: de 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

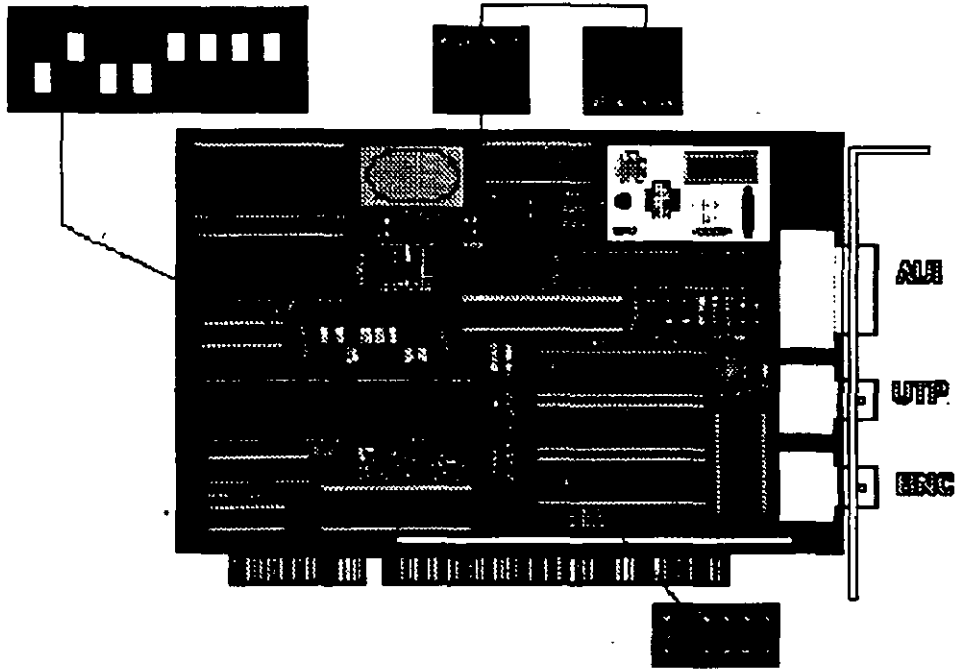


Notas:





ETHERNET



Notas:

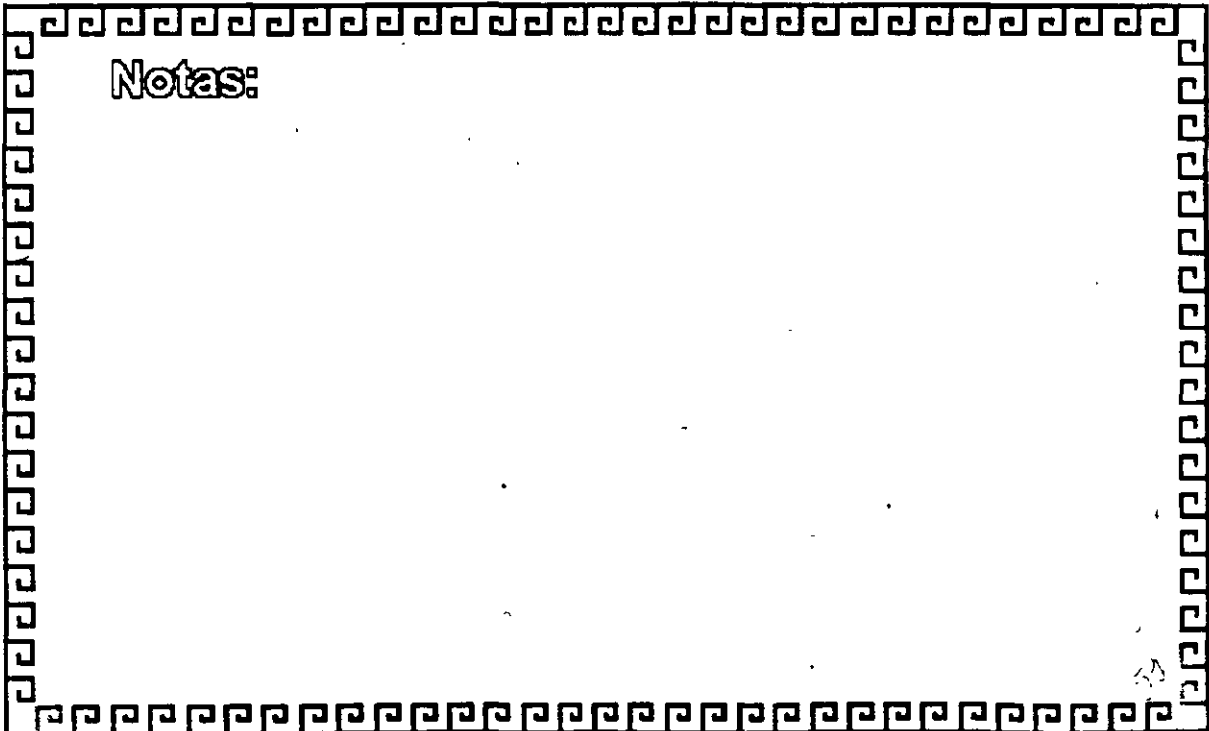
51



ETHERNET

Características

- Creada por XEROX (1970)
- Estandar más Estable
- Versátil en distintos Ambientes
- Instalación Compleja



Notas:



ETHERNET

ESPECIFICACIONES TECNICAS

Velocidad 10Mbits/seg

Protocolo CSMA/CD

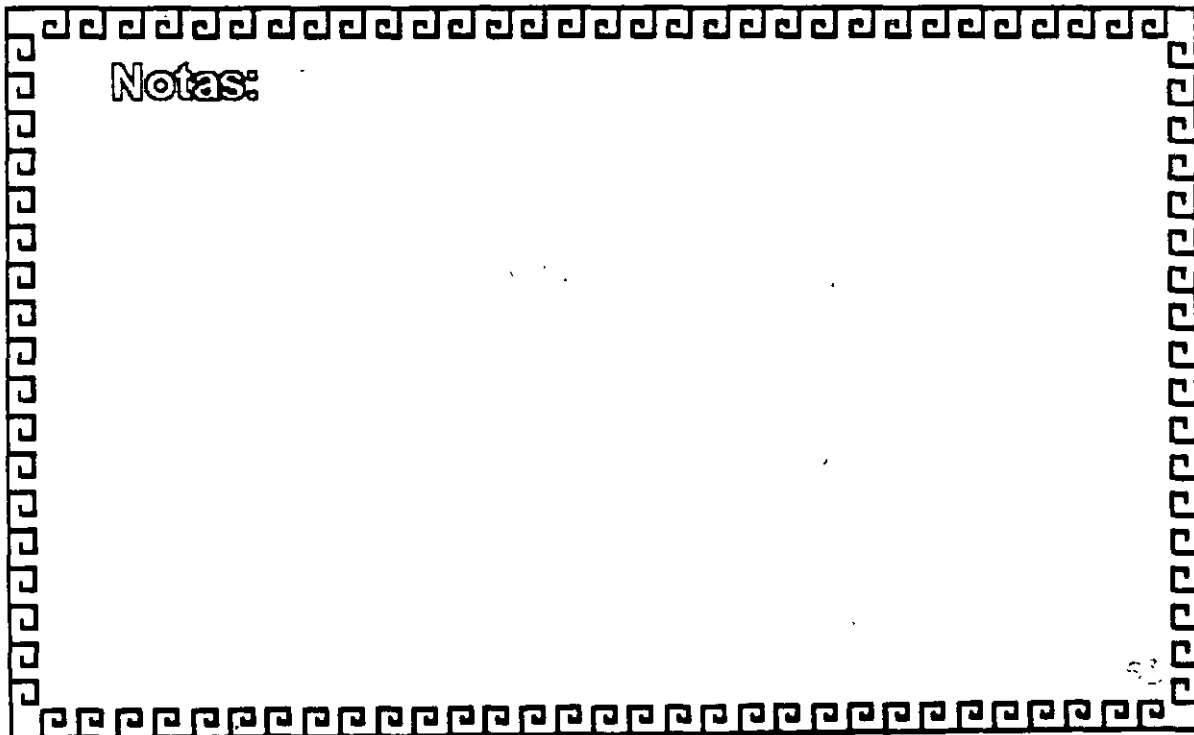
Nodos 1 a 1023

Cableado

THICK (RG-11) 500m*
THIN (RG-58) 300m*
TWISTED PAIR 150 m
FIBRA OPTICA

* Máximo 3 segmentos

Notas:



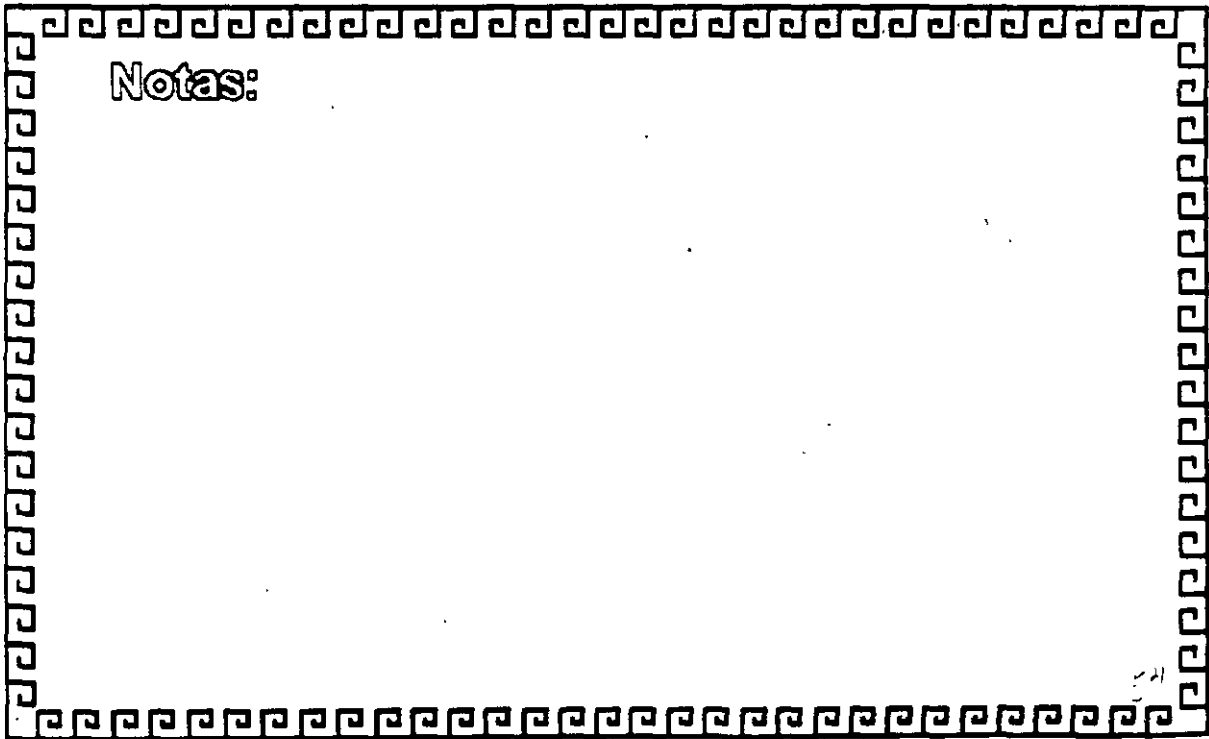


ETHERNET

FABRICANTES MAS IMPORTANTES

- 3 COM
- EXCELAN
- MICRON
- NOVELL
- GATEWAY
- S M C
- INTEL

Notas:





ETHERNET

VARIANTES EN INTERFACES PARA PC's

- Tamaño de BUFFER 8, 16, 40, 64 Kbytes
- Bus de 8, 16, 32 Bits o Microcanal
- Uso de D M A
- Procesador
- Generación: 1ra. 2da. y 3ra.

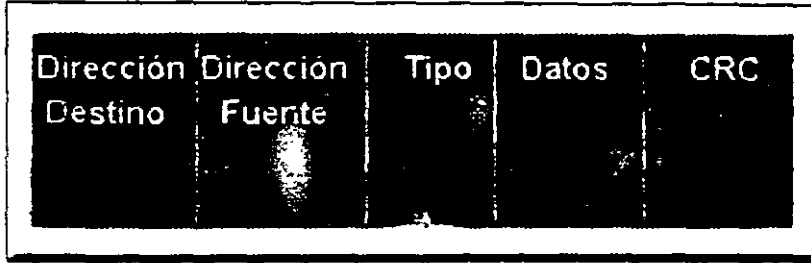


Notas:

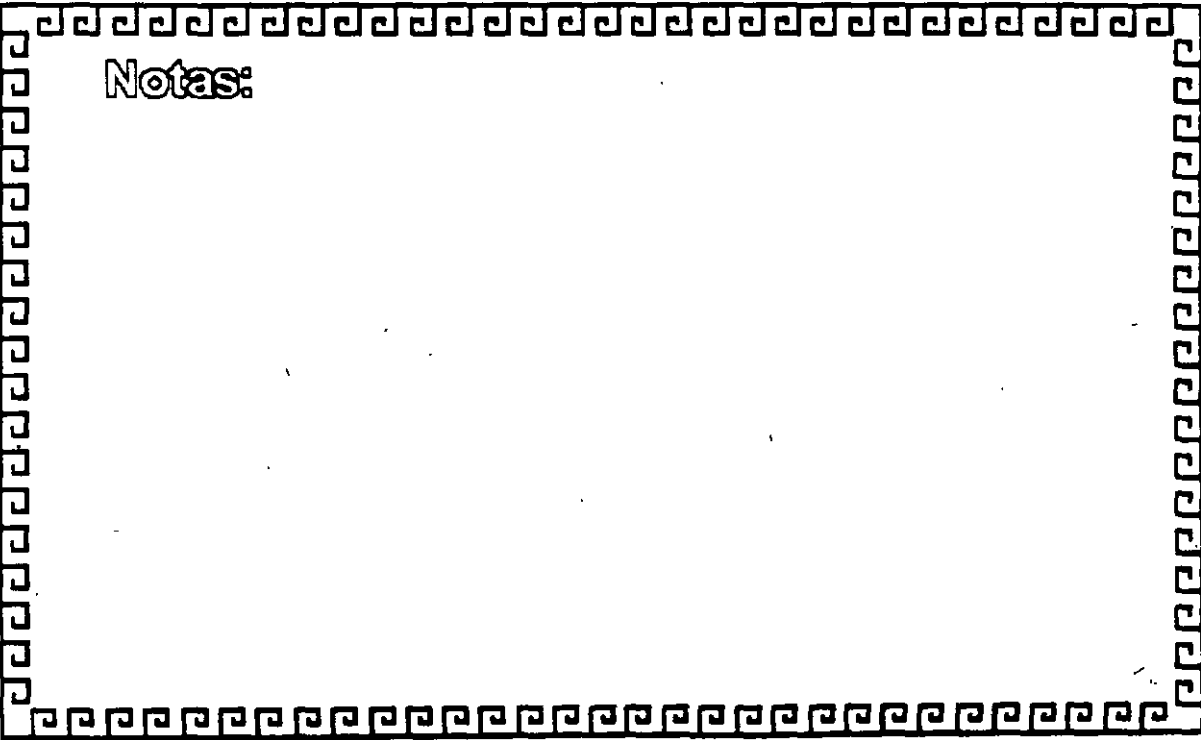


ETHERNET

FORMATO DEL FRAME



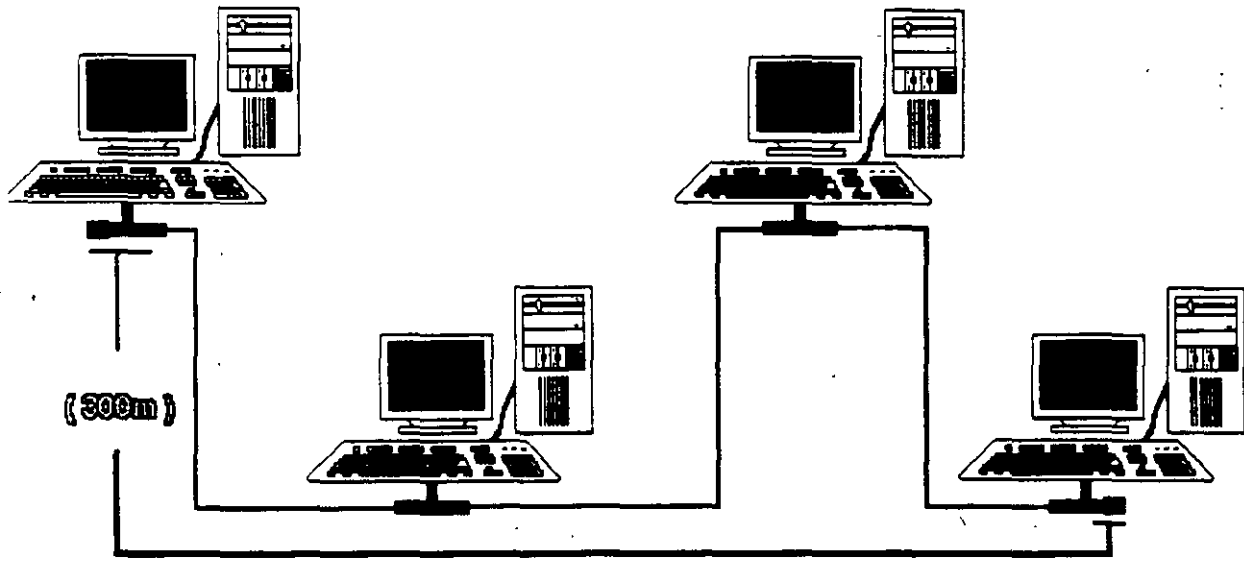
Notas:





ETHERNET

CONFIGURACION TIPO EN THINLAN



Notas:



ETHERNET

Resumen Técnico

- Velocidad 10 Mbits/seg.
- Estandar más utilizado en el orbe
- Más alto rendimiento (performance)
- Cableado
 - Coaxial Delgado (300 m/seg.)
 - Coaxial Grueso (500 m/seg.)
 - Par telefónico (150 m/seg.)
 - Fibra Optica
- Conectividad hacia otros sistemas
- Norma 802.3 (IEEE)

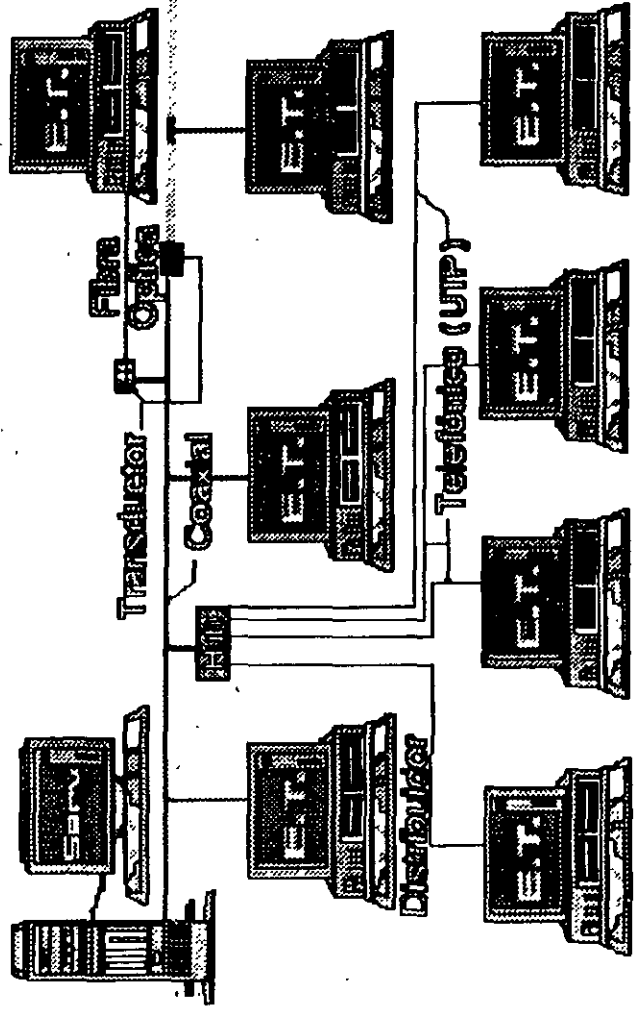


Notas:





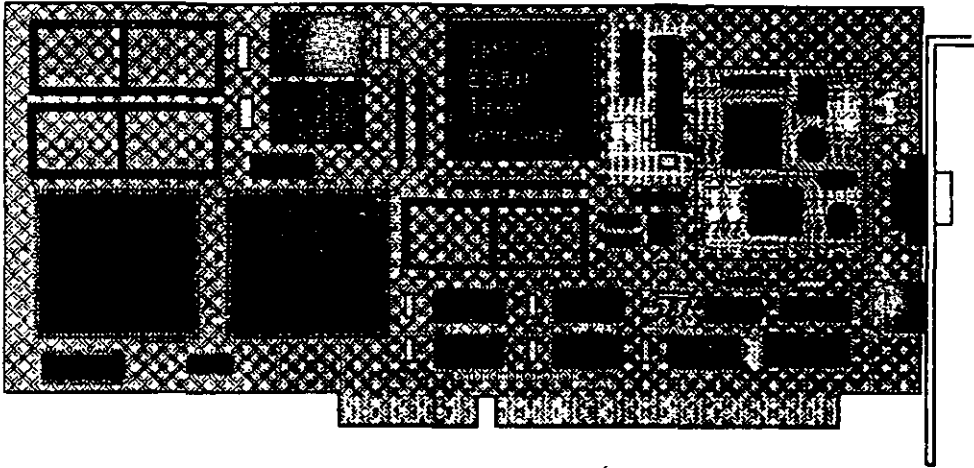
ETHERNET



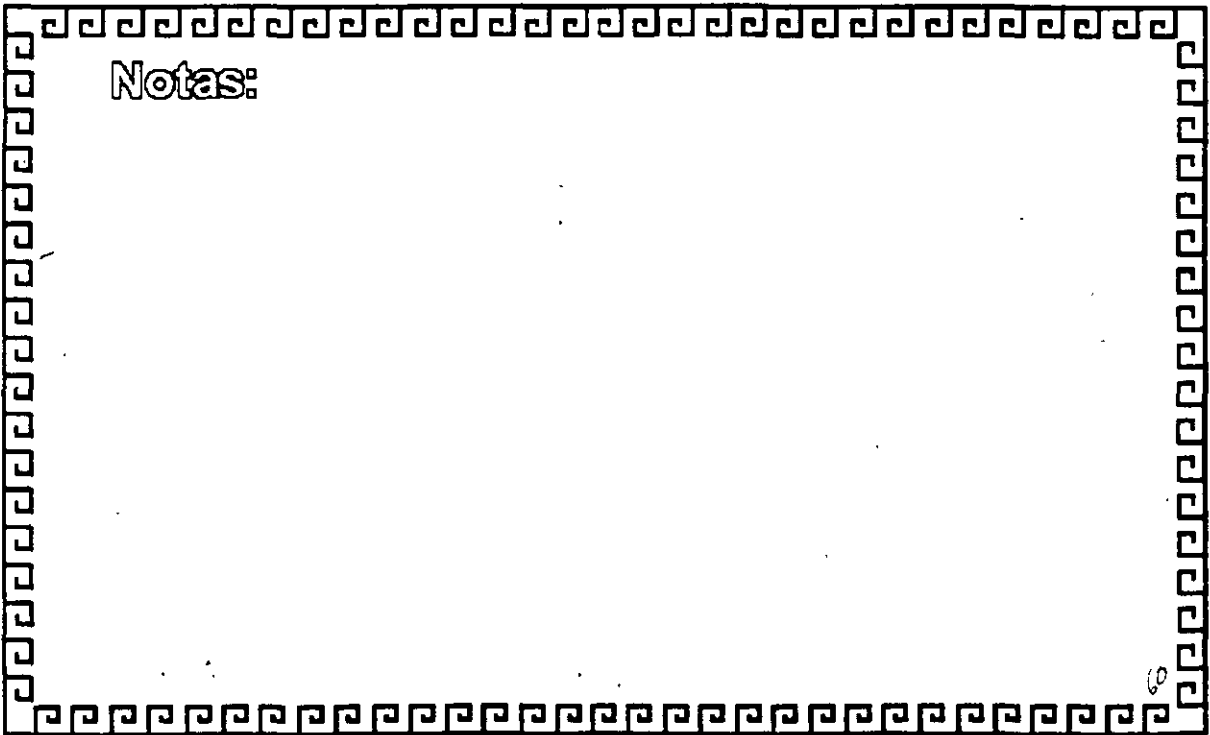
Notas:



TOKEN RING



Notas:





TOKEN RING

CARACTERISTICAS

- Creada por IBM
- Alta Conectividad en IBM
- Buen complejo
- Buen rendimiento
- Opción de 4 y 16 Mbits/seg.



Notas:





TOKEN RING

Especificaciones Técnicas

Velocidad 4 ó 16 Mbits/seg.

Protocolo Token Passing

Nodos 1023

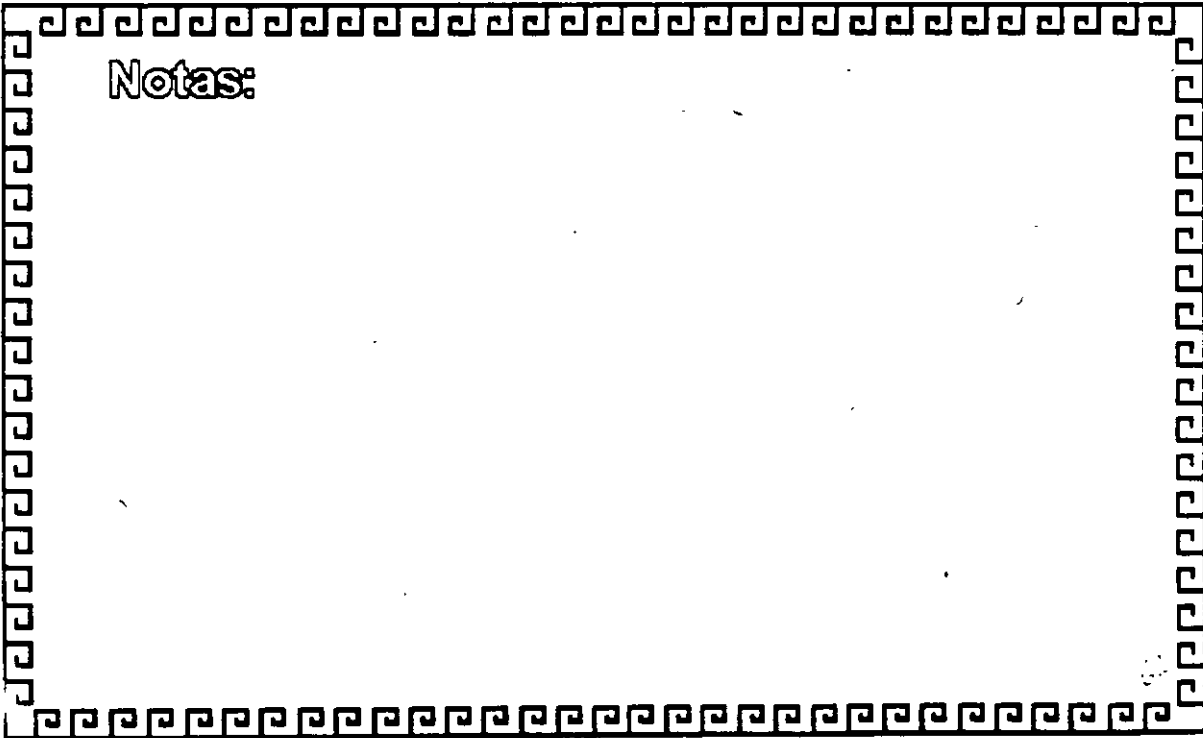
Instalación ... MAU's

Cableado

STP/IBM tipo 2

UTP

FIBRA OPTICA



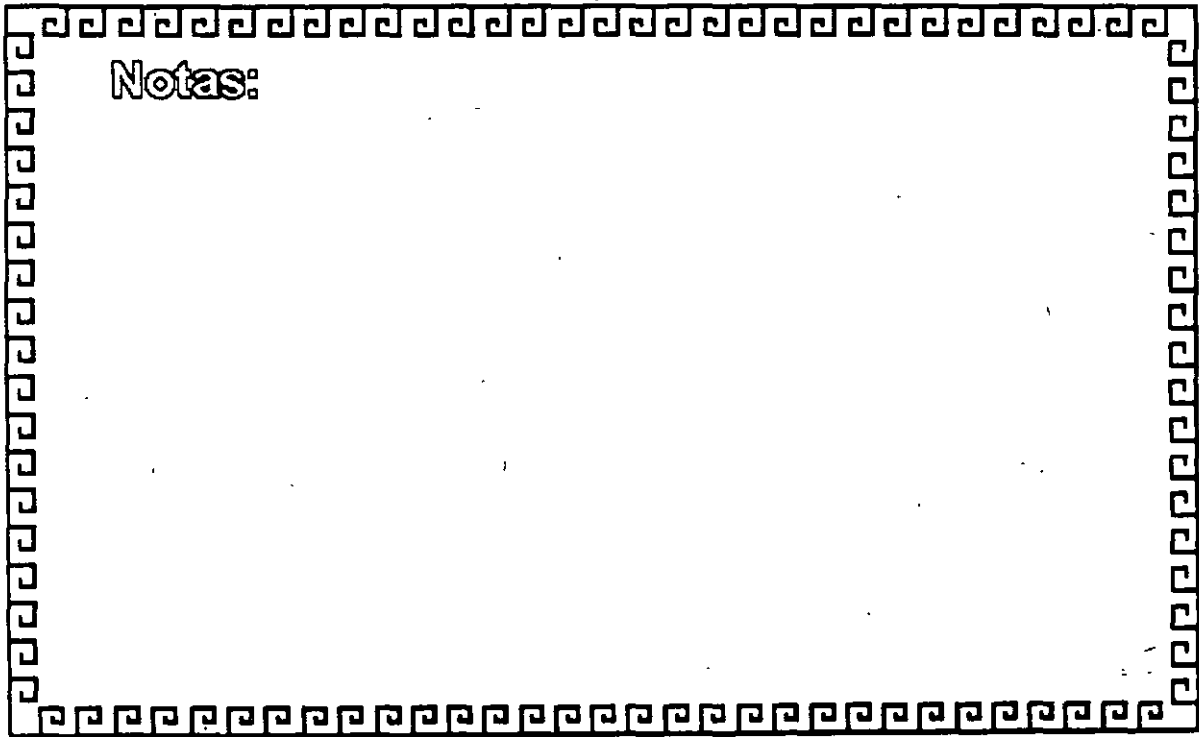
Notas:



TOKEN RING

FABRICANTES MAS IMPORTANTES

- 3 COM
- IBM
- MICRON
- UNGERMAN - BASS
- PROTEON



Notas:



TOKEN RING

FABRICACION

El conjunto de Chips para Token Ring se desarrolló conjuntamente entre IBM y Texas Instruments. Casi todas las interfaces Token Ring se basan en el Chipset de T.I. (TMS380)

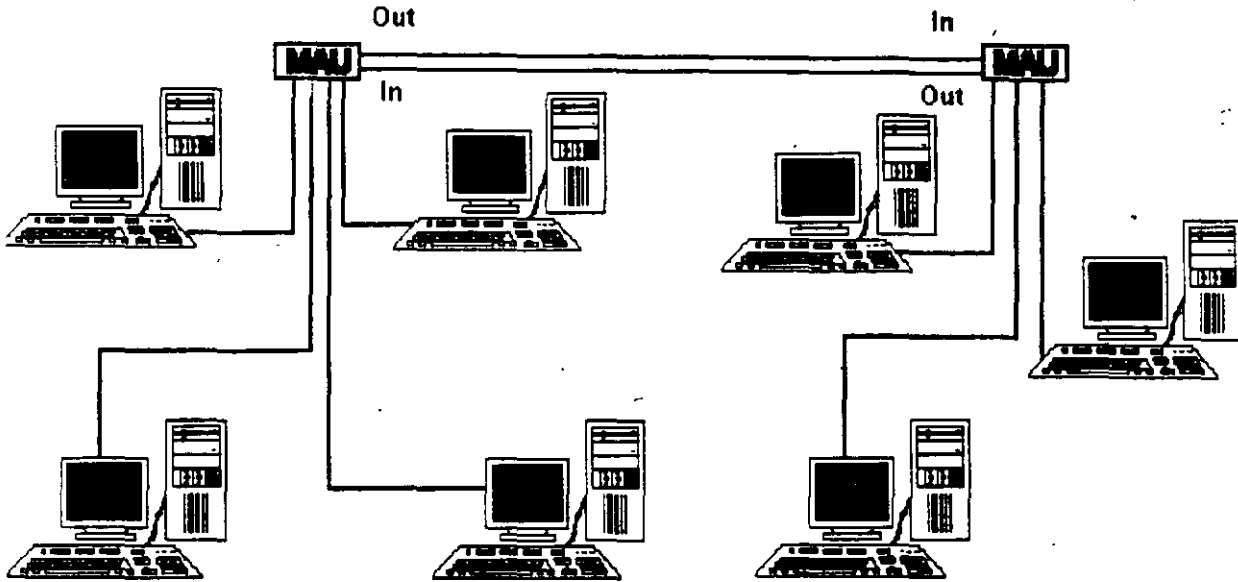


Notas:



TOKEN RING

CONFIGURACION TIPO TOKEN - RING




Notas:



TOKEN RING

Resumen Técnico

- 4 Mbits/seg.
 - Topología de Estrella distribuida
 - Norma 802.5 (IEEE)
 - Protocolo Token Passing
 - Cable IBM tipo 2
 - conectividad hacia ambiente IBM
- 



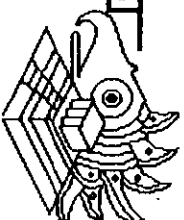
Notas:

INSTALACIÓN Y MANEJO DE REDES (LAN) EN PLATAFORMA UNIX

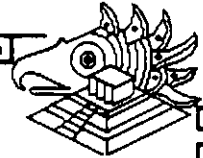
3.- FABRICANTES DE UNIX



Mayo de 1996.



3. FABRICANTES DE UNIX



SCO

UNIX

DE

SANTA CRUZ OPERATION



SCO UNIX

Implementación del sistema UNIX en la
plataforma Intel 386 y 486, liberado en 1989.

Basado en AT&T System V 3.2 Primera Licencia "UNIX"

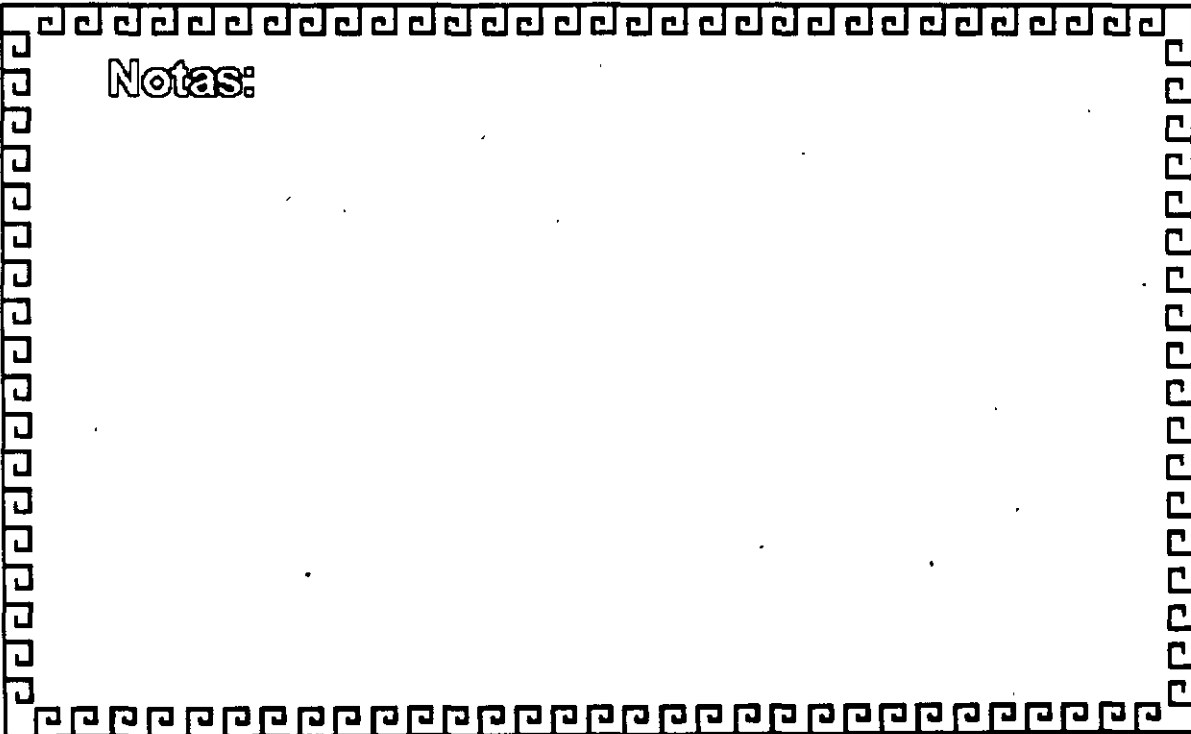
Incluye extensiones de BSD 4.3 y XENIX

Más del 80 % del mercado UNIX en plataforma Intel

Amplio soporte de fabricantes de hardware y periféricos.



Notas:





SCO UNIX

Más de 4000 aplicaciones de software

El 19% de las Acciones de la Cia. SCO UNIX

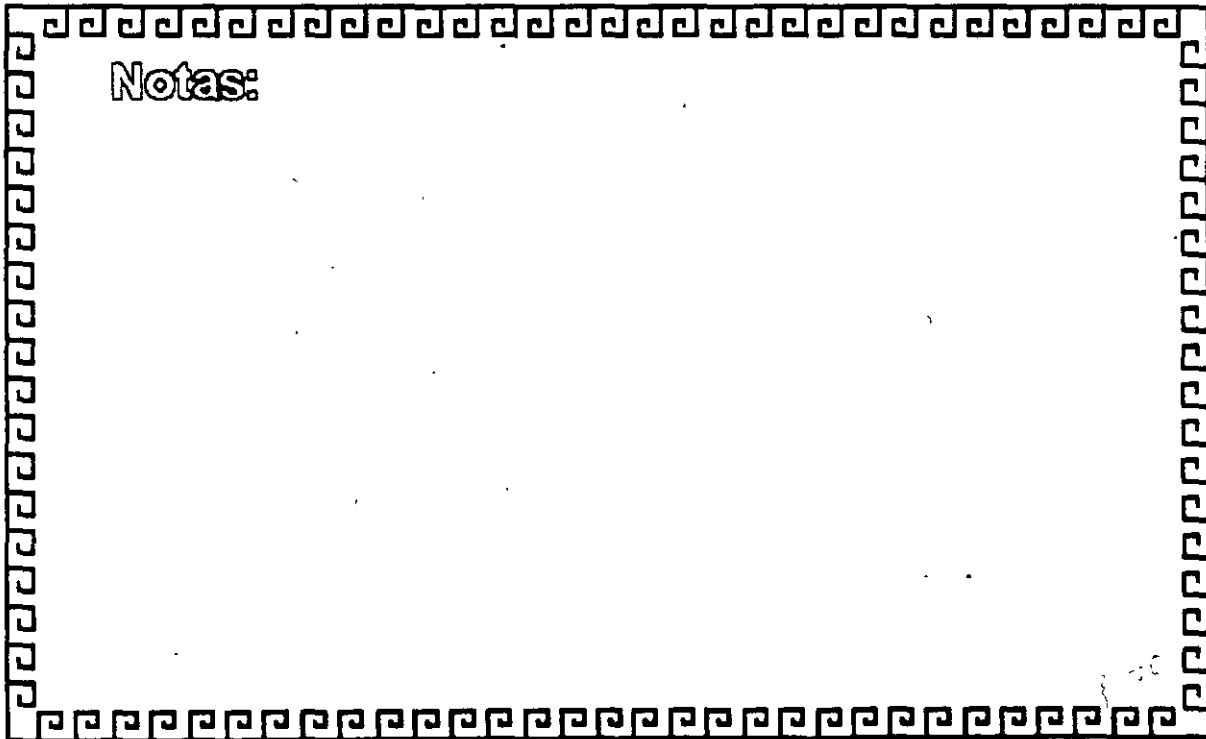
pertenecen a Microsoft

CARACTERISTICAS TECNICAS :

Multiusuario, Multitarea,

Memoria Virtual Paginada

Soporte de Multiprocesamiento



Notas:



SCO UNIX

MULTIUSUARIO MULTITAREA

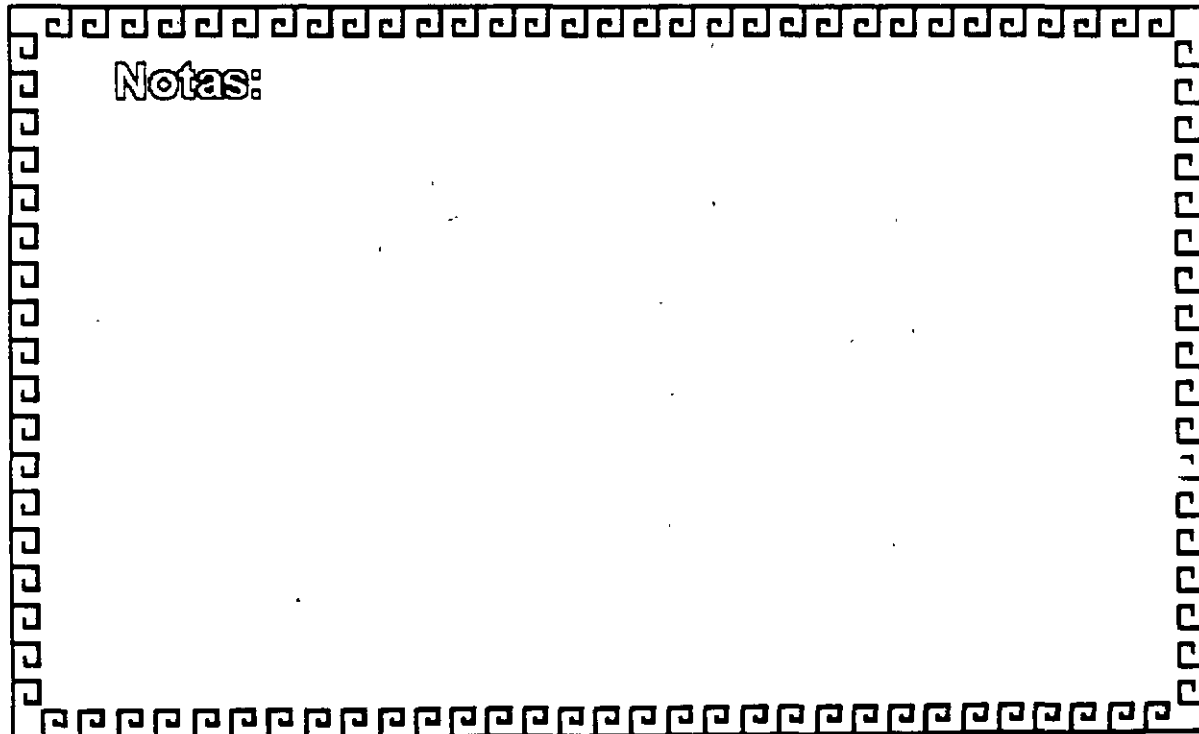
Sistema multiusuario con seguridad y
compartición de recursos
Preemptive multitasking

MEMORIA VIRTUAL

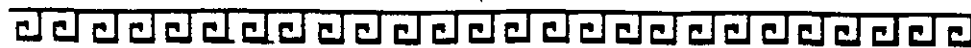
Paginación por demanda de 4 KB.

Soporte de Swaping

(intercambio de procesos a memoria secundaria)



Notas:



SCO UNIX

Basado en el EAFS (Enhanced Acer File System)

Soporte de nombres largos (256 caracteres)

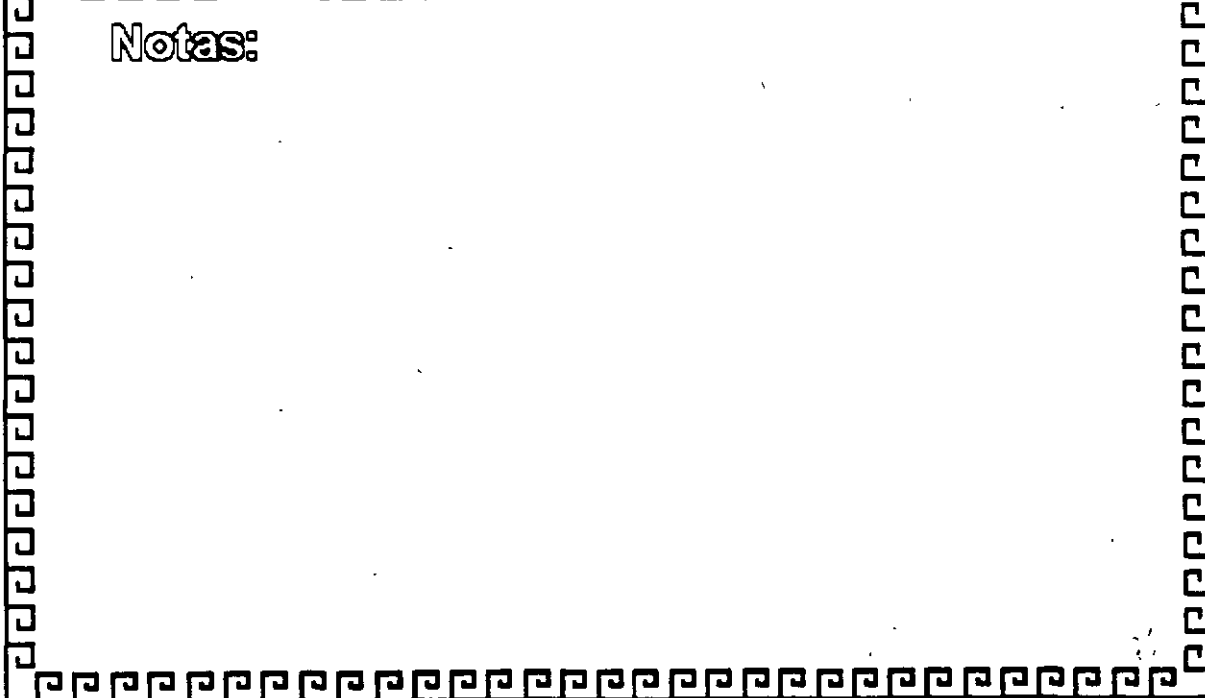
Soporte de ligas y ligas simbólicas

Soporte de DOS, XENIX Y UNIX V

Soporte de CD-ROM (high sierra)



Notas:

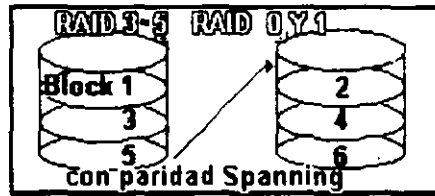
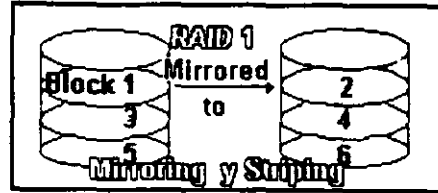
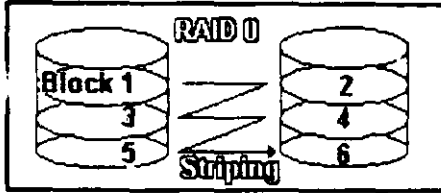




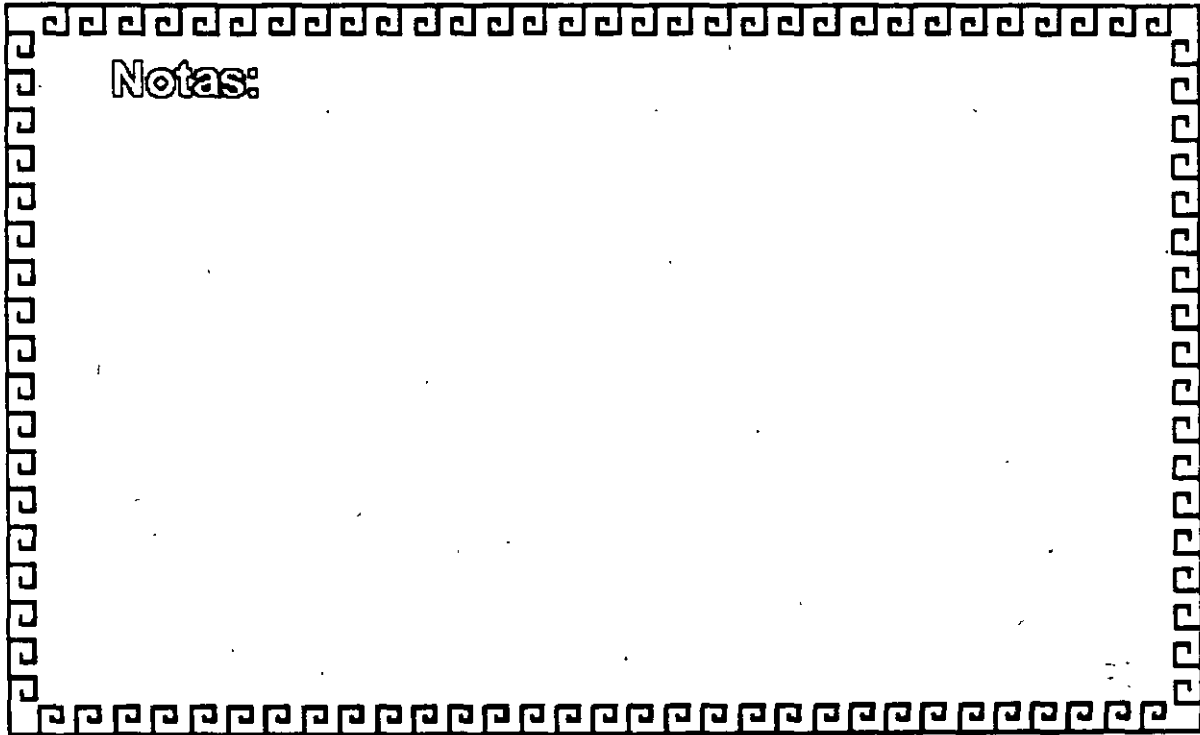
S@O UNIX

SOPORTE DE RAID

(Redundant Arrays of Inexpensive Disks)



Notas:





SCO UNIX

PORTABILIDAD

X / Open XPC93

IEEE Portable Operating System

Interface Specification (POSIX) 1003.1

FIPS 151-1 Extención a IEEE POSIX

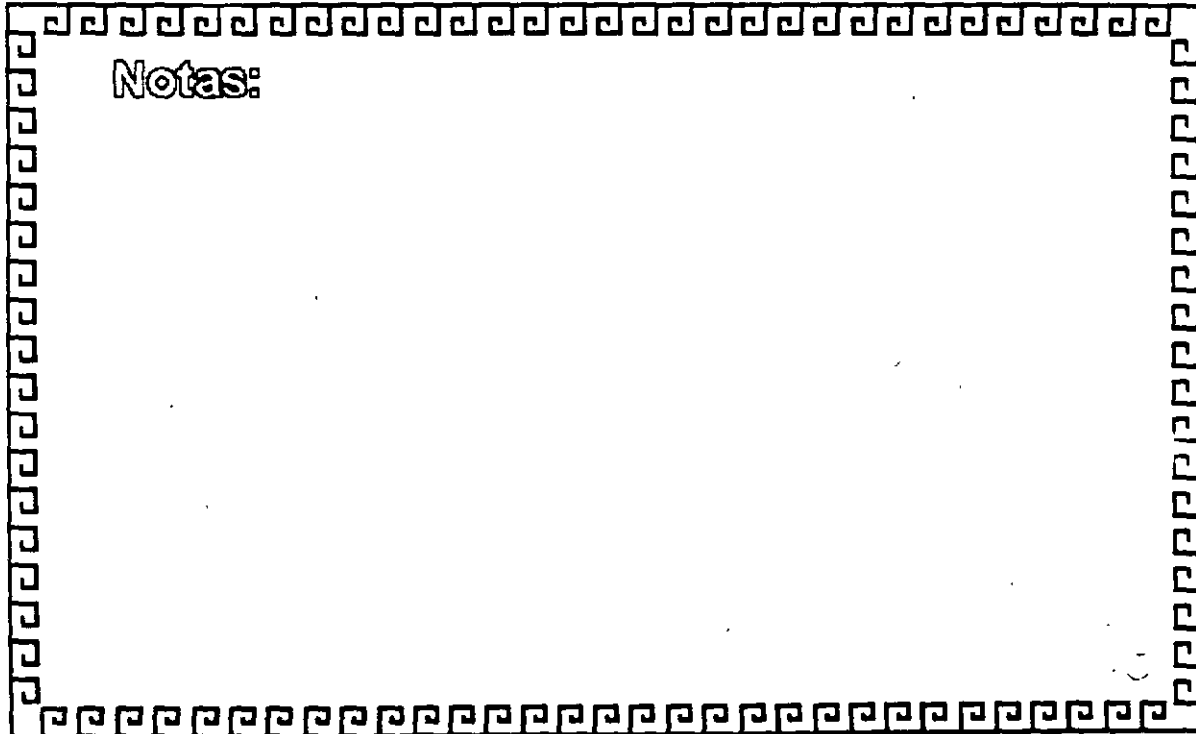
AT & T System V Interface Definition (SVID) Issue 2

Intel 386 Family Binary

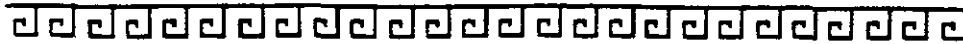
Compatibility Specification

Edition 2 (IBCS-2)

ANSI X3.159 1989 (C Language)



Notas:



SCO UNIX

SEGURIDAD

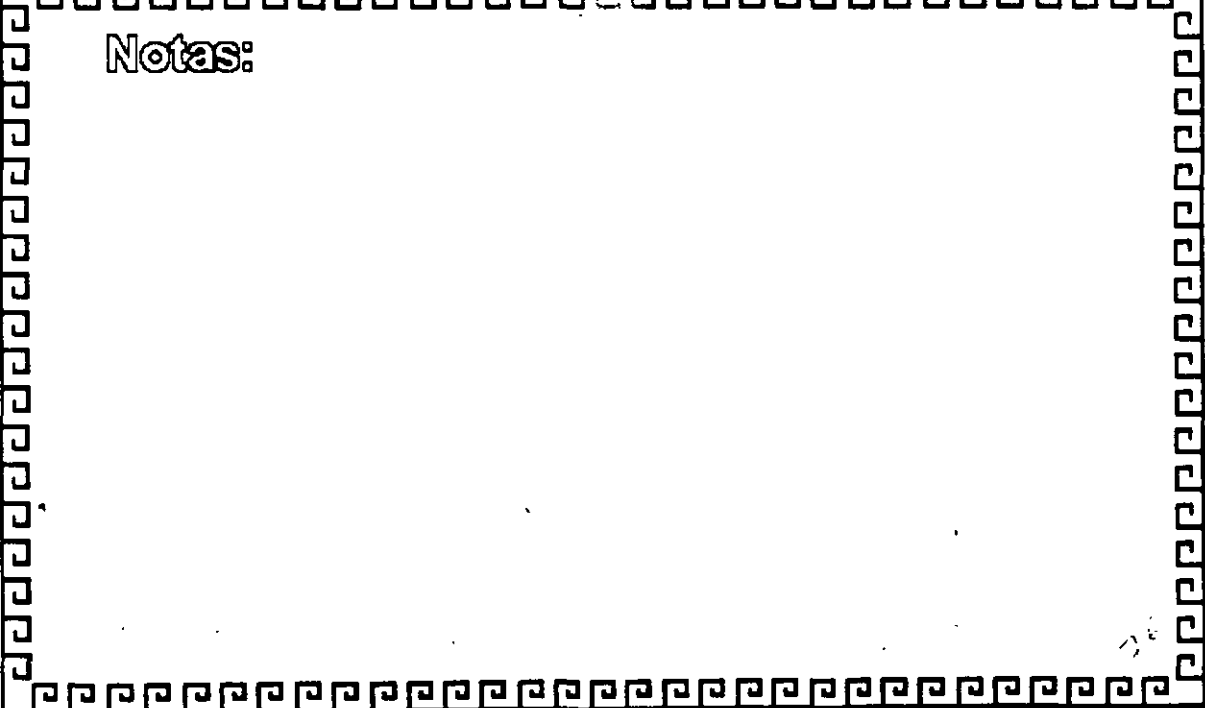
DoD TCSEC Nivel C2
Nivel C2 mejorado (B1 parcial)
Nivel B1 (terceros)

NETWORKING CON SCO

UUCP (tradicional)
X.25
TCP / IP
OSI
SPX / IPX
NFS



Notas:





SCO UNIX

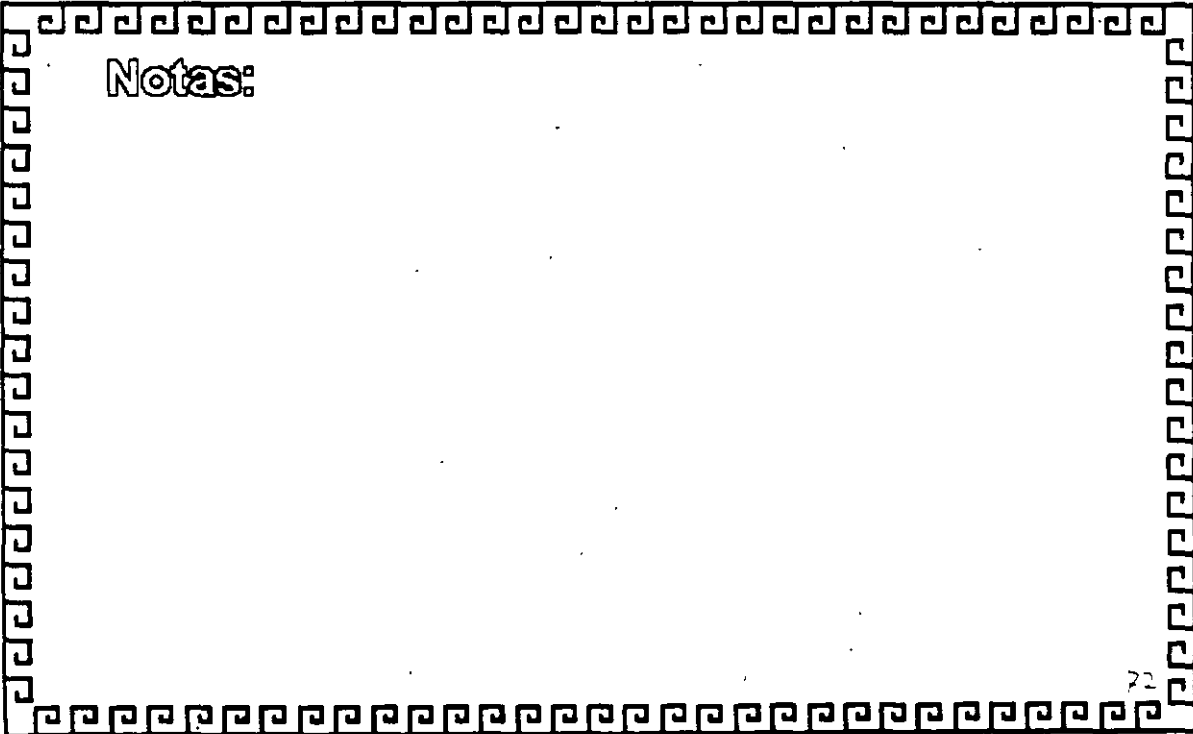
TCP / IP

Permite ligar redes heterogéneas
Desarrollado por ARPA (DoD)
Estándar de facto en el mundo UNIX

LAN MANAGER EN UNIX

Interface de NetBIOS para TCP / IP
NETBEUI
Compatibilidad con NFS

Notas:



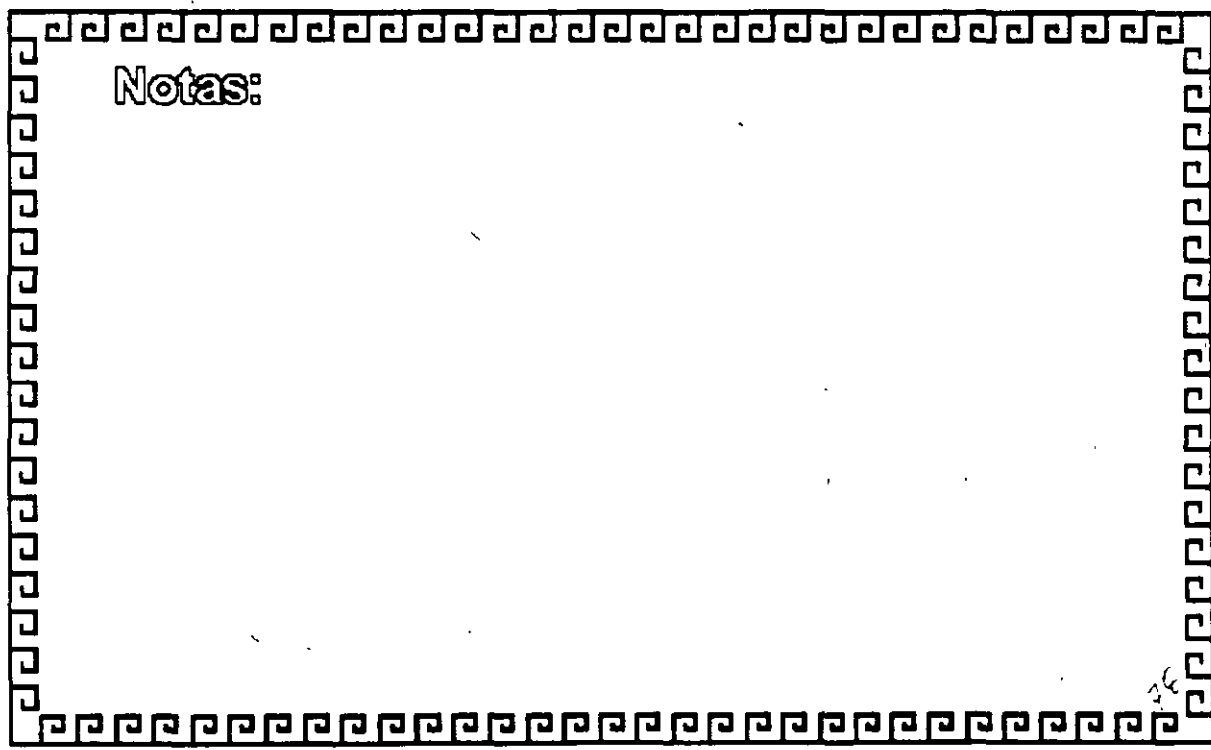


UNIXWARE

DE

UNIVEL

Notas:



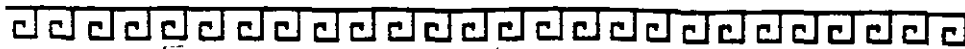


QUIEN ES UNIVEL

- * Participación de Novell y USL
- * Diseñado con los recursos de estas dos Empresas
- * Univel se encuentra en:
San Jose, C A
Sandy, U T
Summit, N J
- * Se estableció en diciembre de 1991



Notas:



UNIXWARE

- **Construido con la potencia del Sistema UNIX**
- **Proporciona facilidad de uso**
- **Aprovecha la sinergia entre los sistemas UNIX y Netware**
- **Aprovecha los recursos de Novell para construir de UNIX un canal fuerte de distribución**



Notas:



CRECIMIENTO DE MERCADO Y CONVERGENCIA

Networked PCs ○ ○ Technical UNIX 1985

Workgroups ○ ○ UNIX 1990

Network Computing ○ ○ 1995



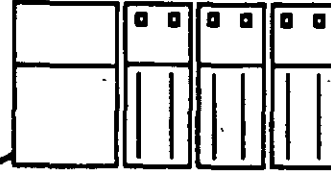
Notas:

EL AMBIENTE DE LOS USUARIOS DEL RIGHTSIZING



UPSIZING

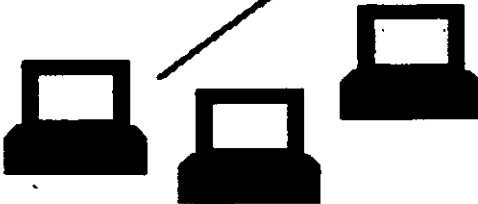
- Gran ejecución y funcionalidad
- Acceso a los datos de la Corporación y a los servicios del hardware
- MIS obtiene control
- Explora nuevas tecnologías del hardware



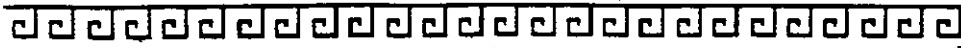
Marco de
Computación

DOWNSIZING

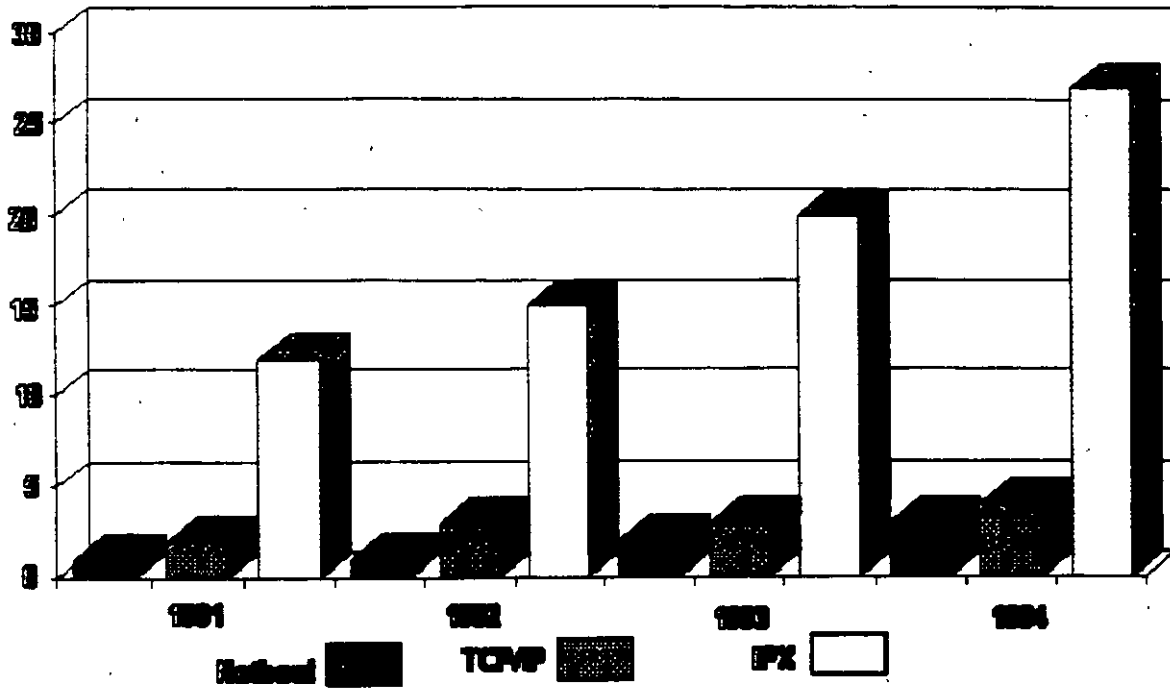
- Explora la tecnología del hardware
- Sistemas Abiertos
- Ambiente poderoso de desarrollo
- Adaptación cliente/servidor
- Reducción del costo



Notas:



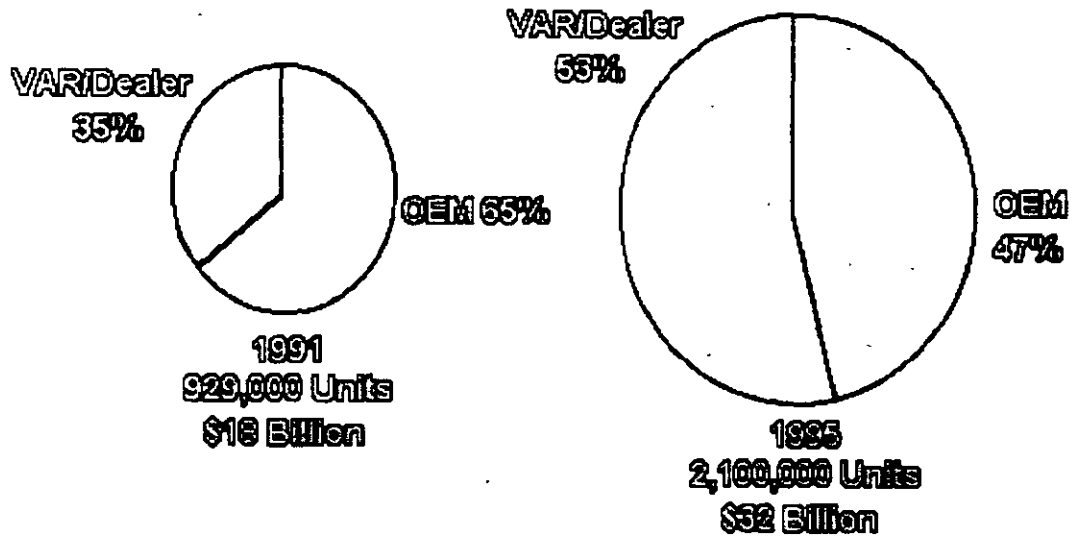
NODOS DE RED INSTALADOS



Notas:



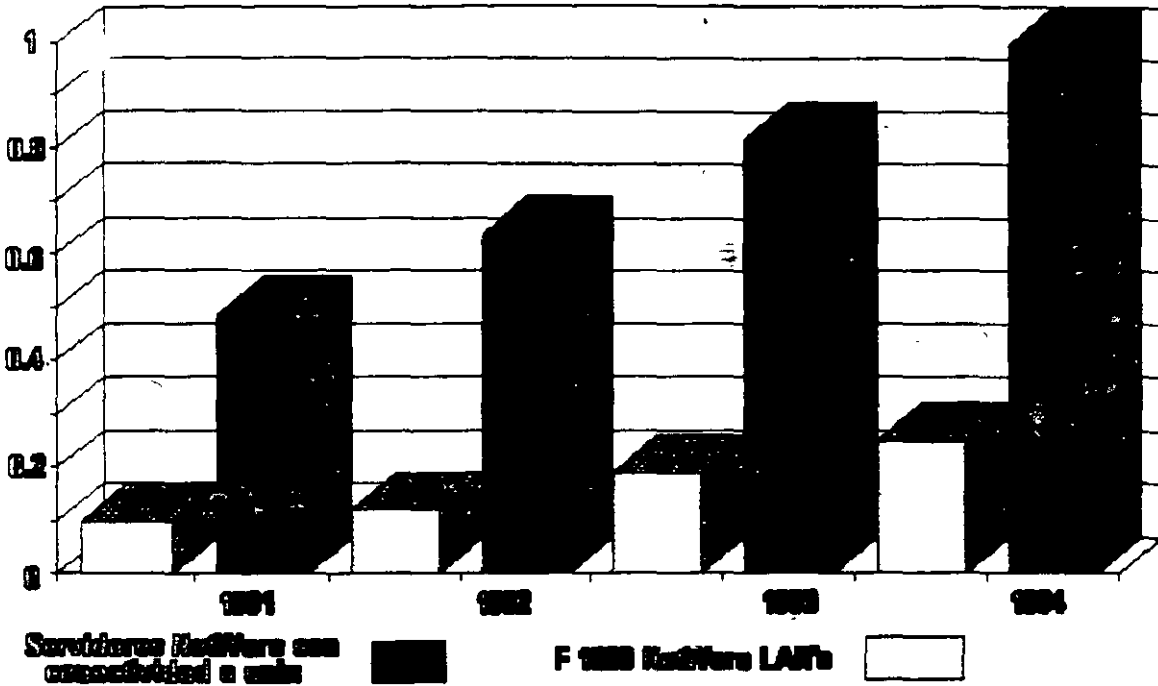
DISTRIBUCION DE UNIX



Notas:



EMPRESAS F 1000 CON SOPORTE UNIX



Servidores RedWare con consolas en units F 1000 RedWare LAN's

Notas:

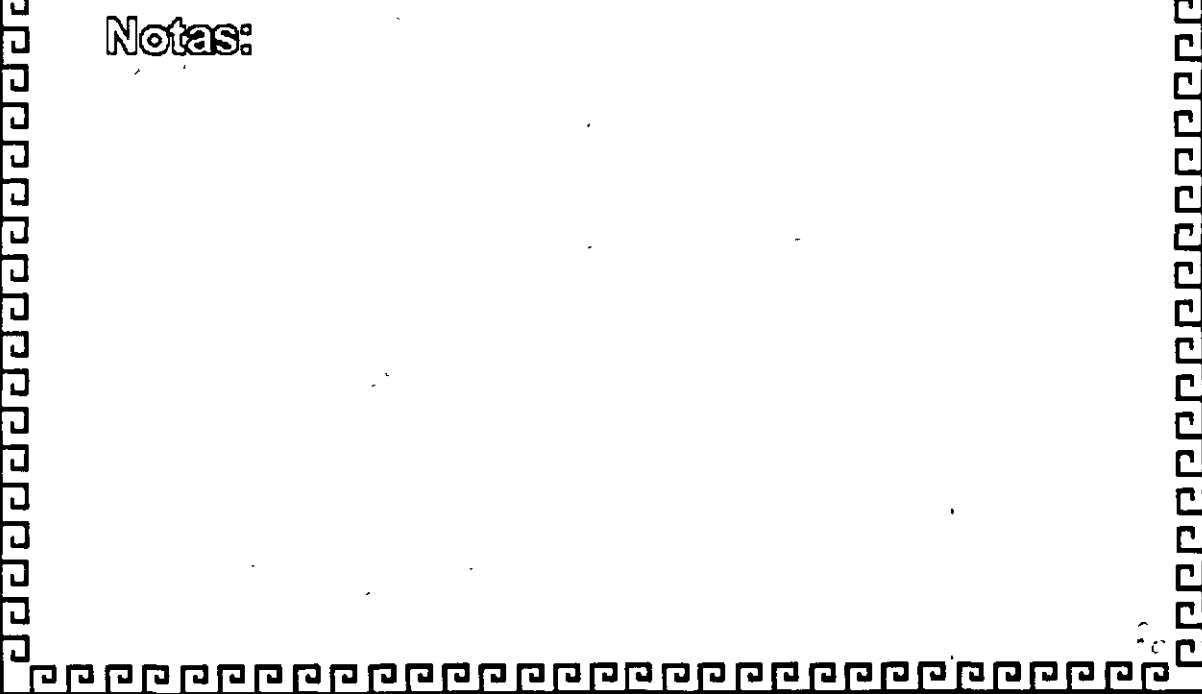


CARACTERISTICAS DEL PRODUCTO

- * Última tecnología del Sistema UNIX
- * Basado en estándares
- * Cliente/Servidor
- * Fácil de usar
- * Integración con Netware
- * Compatible con el Hardware estandar de la Industria



Notas:



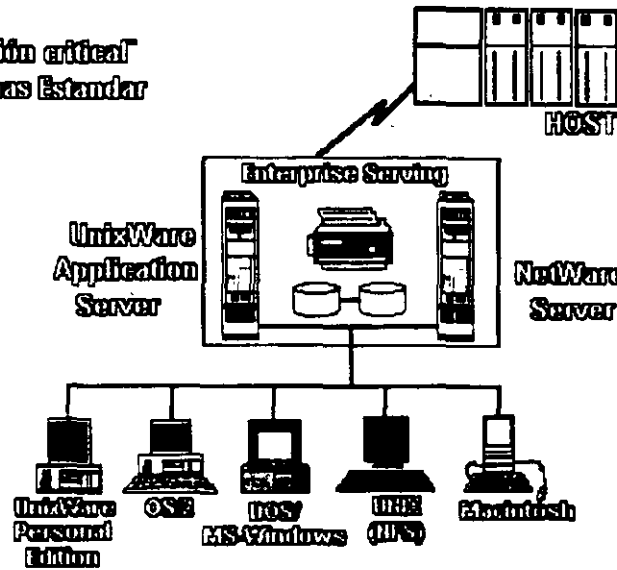
SOLUCION CLIENTE/SERVIDOR



UNIX SVR4.2: Poderoso, robusto, "misión crítica"
 - Flexible, Cliente/Servidor, Plataformas Estandar

Integración con NetWare
 - Acceso a los servicios de la Red y mantiene un ambiente heterogéneo para las PCs

Distribución de Novell para:
 - Integra soluciones de llave en mano
 - Servicio y mantenimiento
 - Enseñanza y capacitación
 - Existencia local



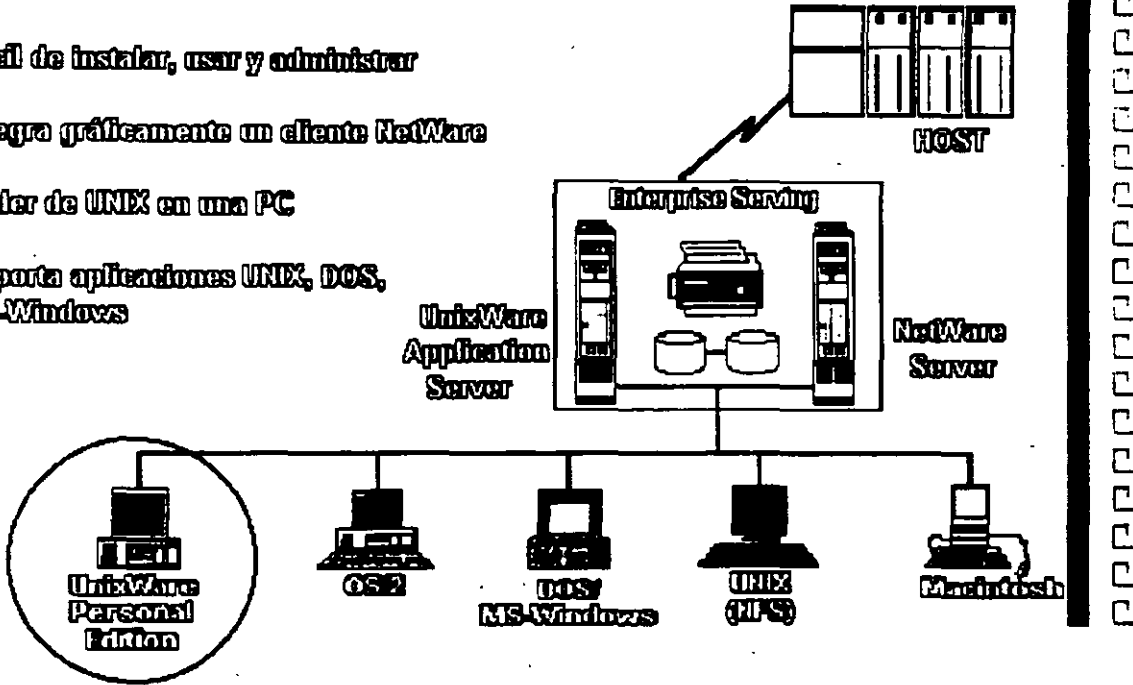
Notas:

27

POTENCIA DESKTOP UNIX



- ' Fácil de instalar, usar y administrar
- ' Integra gráficamente un cliente NetWare
- ' Poder de UNIX en una PC
- ' Soporta aplicaciones UNIX, DOS, MS-Windows



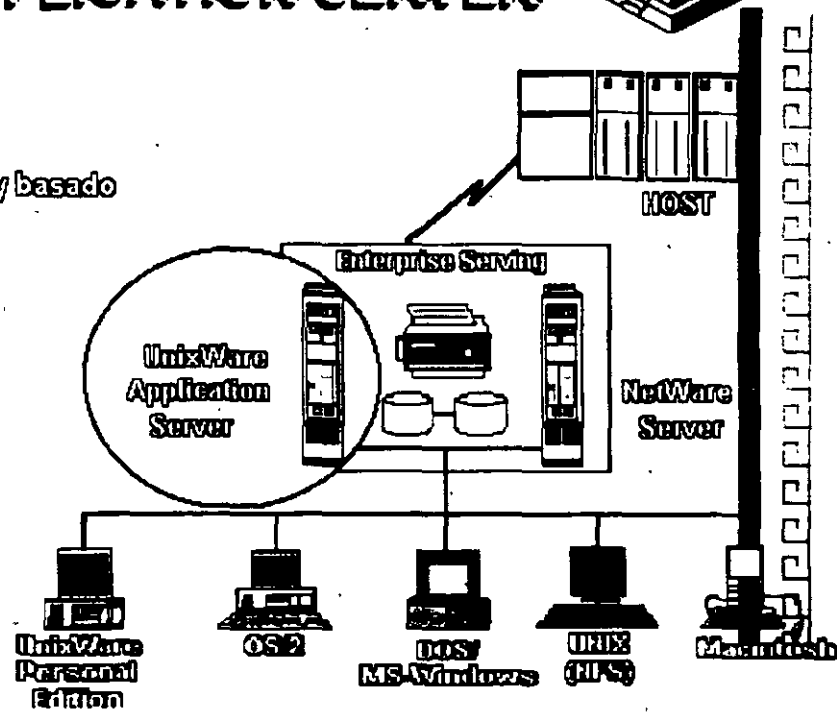
Notas:

92

UNIXWARE APPLICATION SERVER



- Multiusuario: X Windows y basado en caracteres
- Tolerante a fallas
- Manejo Gráfico
- Integración con NetWare
- TCP/IP y NFS
- Rápida instalación sobre el Servidor



Notas:

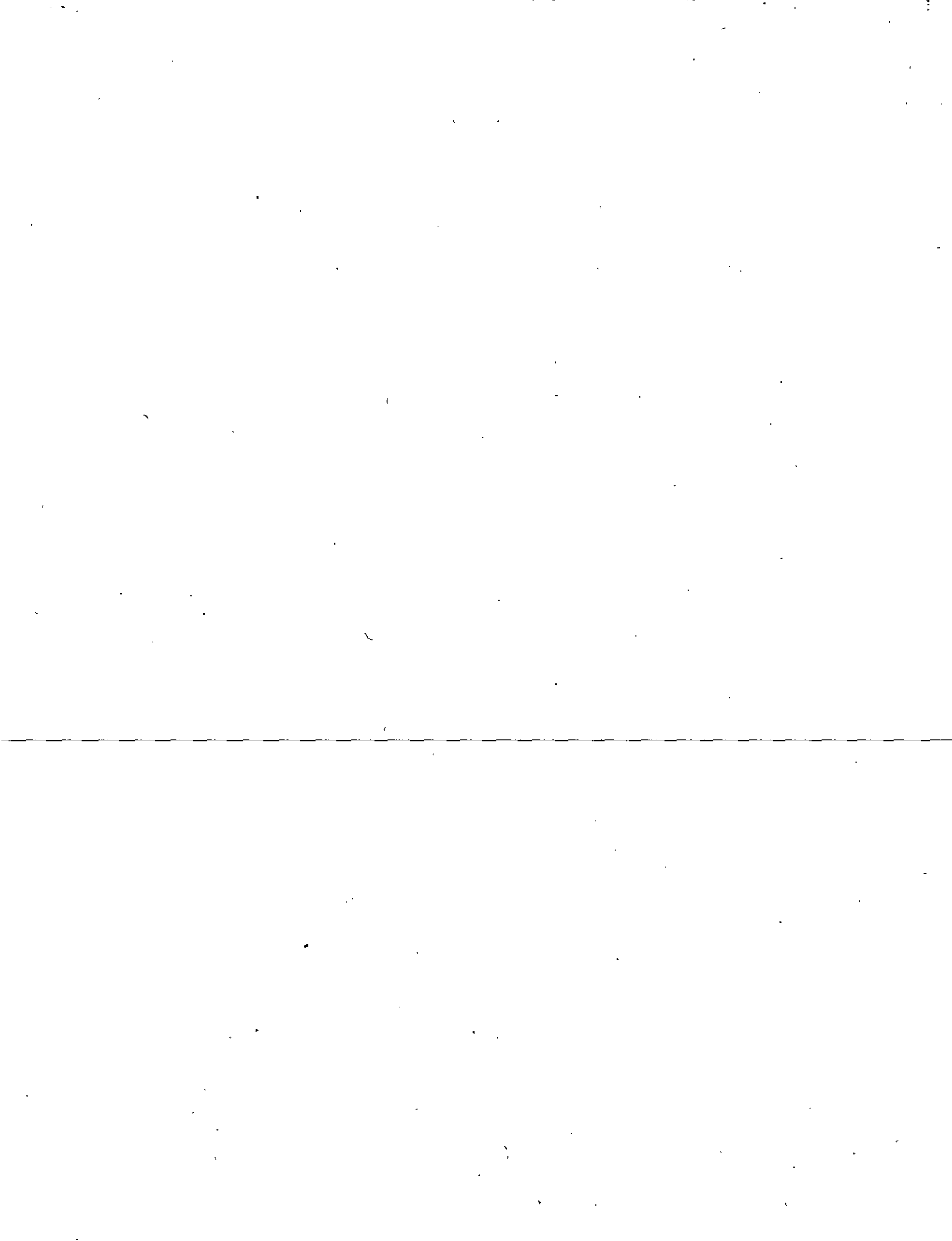


INSTALACIÓN Y MANEJO DE REDES (LAN) EN PLATAFORMA UNIX

4.- INSTALACIÓN DE UNIX



Mayo de 1996.



Chapter 7

Working from the command line

In Open Desktop, you can control your computer by clicking on or dragging icons and by selecting actions from menus. Such graphical environments (also called *graphical user interfaces* or *GUIs*) were not widely used until computers with the required memory and speed became generally available.

Prior to the development of GUIs, most communication with computers was through the *command line*. A command line is a line on the screen on which you type commands (instructions) to your computer. It is usually identified by a symbol such as “%” or “\$”, called a *prompt*:

❖ type command here

A command line interface is easier for the computer because it only needs to read and react to one line on the screen. It is more difficult for the person using the computer because it requires that obscure commands be remembered and entered in a precisely prescribed way. The command line interface remains a powerful tool because it provides direct access to operating system and networking functionality, and because command line scripts can be used to automate and customize routine tasks.

NOTE The rest of this book is about working from the command line. If you do most of your work on the Desktop, you can skip Chapters 7 through 24 until you need to know how to interact directly with the operating system or networking software beneath the Desktop.

Operating systems

The Open Desktop graphical environment provides a simple, intuitive way for you to work with your computer. Beneath the Desktop, hidden from view, is a complex package of sophisticated software. Your computer does not understand icons and scroll bars; it operates in a language made up entirely of 0's and 1's. Between that binary language and the graphical environment are many layers of software, each more sophisticated than the one it is built upon.

An *operating system* is a group of programs that provide basic functionality on a computer. These programs operate your computer hardware in response to commands like **copy**, **sort**, and **print**. Applications and other programs can use these commands without worrying about the specific signals that make a disk drive work on a particular computer. Because the operating system takes care of such low-level concerns, programs can be more portable and easier to write. An operating system can be seen as a set of functional building blocks upon which other programs depend. It also manages computer resources and resolves resource conflicts, as when two programs want to use a disk drive at the same time.

Open Desktop is constructed on top of the UNIX operating system. The UNIX system is used on a wide variety of hardware, ranging from personal computers to supercomputers. It is characterized by its rich assortment of basic tools (or *utilities*), and by its ability to support multiple users running multiple programs at the same time. Programs written to run on UNIX operating systems will run on Open Desktop. See Chapters 8 through 13 for information about working with UNIX commands and files.

Another widely used operating system is DOS. DOS was designed to support a single user running one program at a time on a single personal computer. Open Desktop can also run most DOS programs. It does this by translating the DOS commands into equivalent UNIX commands. As discussed in the following chapters, you can use either UNIX or DOS commands from the Open Desktop command lines, and you can read files from either DOS or UNIX disks. See Chapters 14 through 18 for information about working with DOS commands and files. In addition, see "Using DOS utilities" (page 99) for information about special UNIX commands for working with DOS files.

NOTE Although Chapters 8 through 13 provide a basic introduction to working with the UNIX operating system, Chapters 14 through 18 assume you are already familiar with DOS. The DOS chapters focus on using DOS with Open Desktop and the UNIX operating system.

Table 7-1 Similar DOS and UNIX commands

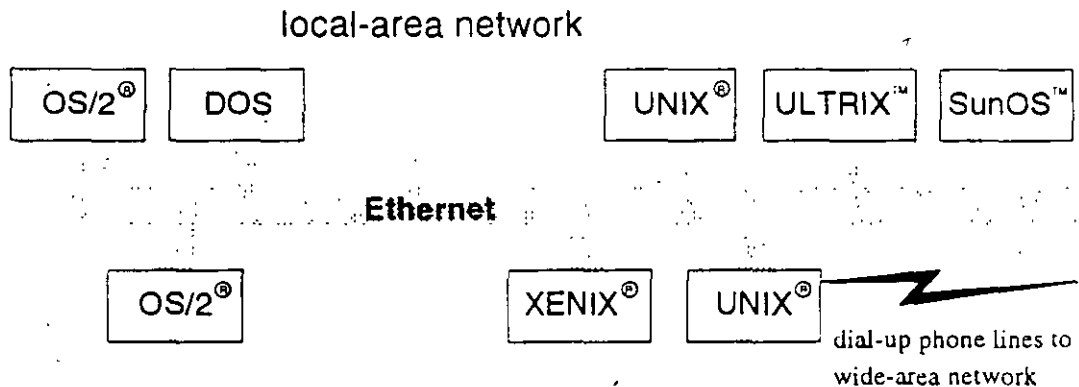
DOS	UNIX system	Action
attrib	chmod	set file attributes (properties, permissions)
cd	cd	change directory
chkdsk	badtrk, fsck	scan disk for errors
cls	clear	clear screen
command	sh, csh, ksh	start a new command processor (shell)
comp	cmp, diff	compare two files
copy	cp, copy	copy a file
date	date	display system date
del	rm	remove a file
dir	ls -l, l	show a long list of filenames
dir/w	lc	show a list of filenames in columns
diskcomp	diskcmp	compare contents of floppy diskettes
diskcopy	diskcp	copy floppy diskettes
edlin	vi, ex, ed, sed	use simple text editor
erase	rm	remove a file
exit	exit	exit current command processor (shell)
fdisk	fdisk	configure hard disk partition
find	grep, fgrep, egrep	search for a sequence of characters in a file
format	format	format a floppy diskette
mkdir	mkdir	make a directory
mode	stty	view or change port settings
more	more, pg	display a file one screen at a time
print	lp	send a file to the lineprinter
ren	mv	rename a file
rmdir	rmdir	remove a directory
sort	sort	sort input file
time	date	display system time
type	cat	display a file
xcopy	cp	copy multiple files and directories
>>	>>	redirect output
<	<	accept input from a file
		pipe output to another command
\	/	pathname separator

Networks

A *network* is a group of interconnected computers. Each computer on the network acts independently, but can transfer information to and from other computers on the network.

A local-area network (LAN) connects computers at one site directly by a high-speed cable, usually an Ethernet™ cable. A wide-area network (WAN), which can be worldwide, connects computers at different sites by transmitting data over telephone lines.

A network might be arranged like this:



This network connects different types of computers running a variety of networking software into a single computing environment. A network like this lets users share the resources of the whole network, which saves time and can eliminate the need to purchase additional hardware and software.

Using the network, you can:

- log in to another computer and use interactive commands such as vi
- execute commands on another computer
- copy files from one computer to another
- exchange mail messages with users on other computers
- share software between computers
- share printers, hard disks, and other devices with other computers

Before you can use any of the networking commands described in Chapters 19 through 24, you must have the required networking software properly installed and configured. Ask your system administrator which networking software your system uses. For networking installation and configuration instructions, see the *Installation and Update Guide* and *Administering Networking Services* in the *System Administrator's Guide*.

Entering commands

After typing a command, press `<Enter>` to send the command to the computer. For simplicity, typing a command, then pressing `<Enter>` is also referred to as "entering a command."

Before you press enter, you can use the `<Bksp>` key (backspace, sometimes labeled with a left-pointing arrow) to back up over and erase previously typed characters. Other command line editing keys may be available, depending on which shell (page 108) you are using and how your system is configured (see your system administrator for details).

UNIX systems do not use the `` key to delete text, like DOS computers do. Instead, the `` key is used to interrupt programs.

NOTE The DOS and UNIX systems have different conventions for filenames, command options, and wildcards, as discussed in the following chapters.

You can also run most Open Desktop accessories and applications from the command line (most Desktop controls are specific to the Open Desktop environment). The command line names of controls and accessories are given in their glossary entries. See a control's or accessory's manual page for information about using it off the Desktop.

More information about commands

There are hundreds of UNIX commands, and most have many options. Only the most useful commands and options are covered in the following chapters. This book is intended only as an introduction to using the UNIX system. Many books are available on just about any aspect of the UNIX system. For more detailed information about a UNIX command mentioned in this book, see the appropriate manual page. Every UNIX command is thoroughly described in a *manual page* (also called "man page").

7. Working from the command line

To see the manual page for a command, at the UNIX command line type:

man *commandname*

Substitute the name of the command for *commandname* (you can also use the more graphical **xman** command instead of **man**).

To read manual pages from the Desktop, double-click on *User and Administrator Manual Pages* in the Help Library window. Then double-click on the manual page you want to see in the command summary or table of contents. See Chapter 4 for more about using online help.

(A letter in parentheses following a command or filename refers to the manual page section where the command or file is documented. For example, the **man(C)** command is documented in the Commands section of the manual pages.)

For more about DOS commands, see your DOS manual.

Cha
Ge

Ent.

e the

istra-
man-
. See

annual
s, the
annual

Chapter 8

Getting started

Most of your communication with the UNIX operating system is through the Desktop: you point and click on objects, and the Desktop tells the operating system what to do.

To communicate directly with the operating system, you must enter commands in the UNIX window. Double-click on the UNIX icon to open the UNIX window.

When you finish entering commands, you can close the UNIX window by entering the `exit` command on the command line. You can also close it like any other window by selecting `Exit` from the `File` menu.

Entering UNIX commands

To enter a command, type its name at the prompt (usually a symbol such as `%` or `$`) and press `(Enter)`.

Entering a command is like engaging in a dialog with the operating system. You wait until the computer gives you a prompt before entering a command. The new prompt tells you that the operating system has finished processing your previous command and is ready for another.

Most UNIX commands have options to modify their behavior. Many commands also take arguments, on which they act. The command comes first, then the options (which are usually indicated by a `-` in front of them), then the arguments, as in the following example:

```
sort -r myfile
```

In this example, `sort` is the command, `-r` the option, and `myfile` the argument

The option `-r` specifies that the sort is done in the reverse of normal order; "`-`" indicates that the letter "`r`" is an *option*, which sets some sort of special condition for the program. (Different programs have different options.)

The argument `myfile` is the name of the file, the contents of which are to be sorted.

Note the UNIX system is *case sensitive*: that is, it assumes that `sort`, `Sort`, and `Sort` are different commands. Most UNIX commands are all lowercase.

Specifying command input and output

Many UNIX commands require *input* and *output*; that is, some information to read and process, and somewhere to store the results. If you do not tell a command where to find its input and output, it makes assumptions about where to read and write information; it uses the *standard input* and *standard output*. (These are, respectively, your keyboard and your screen, which is why information is read from and written to your terminal unless you tell a command to use another destination.)

In the example, `sort -r myfile` gets its input by opening the file named `myfile`. Because no output destination is specified, `sort` sends its results to the standard output destination, which is normally your screen.

You can redirect commands' input and output by using the symbols "`<`" and "`>`" on the command line, followed by the name of the file to read or write. For example:

```
sort < file1 > file2
```

makes `sort` treat `file1` as its input, and send its output to `file2` (that is, the contents of `file1` are sorted and the results are placed in `file2`.)

If you send the output of a program to a file that already exists, the existing file is deleted and replaced by a new file with the same name, containing the output of your program; that is, the contents of the existing file will be overwritten.

To add the output of a command onto the end of a file (known as *appending* the output), type "`>>`" instead of "`>`". For example:

```
sort < file1 >> file2
```

appends the output from `sort` onto the end of `file2`.

Running a sequence of commands

There are three ways to run a sequence of commands:

- You can run them individually by typing them at the prompt, either on separate lines or on the same line separated by semi-colons; for example:

```
ls
pwd
who am i
or
ls; pwd; who am i
```

- You can use an editor to store them in a script file, then run the script (see "Running command sequences" (page 103) for further details).
- If the commands all operate on the same data file one after another, you can run them as a *pipeline*.

A pipeline is a sequence of commands that run, one after another, on the same data. The output of the first command is sent to the second command via a *pipe*. The pipe is represented by the symbol "`|`". For example:

```
sort filename | uniq
```

`uniq` is a program that reads lines from its input, copies them to its output, and eliminates duplicates; that is, if it reads the same line twice it ignores that line the second time. This pipeline sorts the lines in *filename*, then sends the output from `sort` through a pipe to `uniq`, which then sends its own output to the standard destination (in this case, your screen).

You can stack commands up in a pipe by separating them with a "`|`" symbol, and you can send the final output of a pipe into a file with the "`>`" symbol. For example:

```
sort filename | uniq | wc > words
```

This pipeline creates a file called *words*, containing a count of all the words that occur on non-identical lines in *filename*. (`wc` is a program that counts the number of words, lines and characters in its input.)

Aborting a command

Press the `` key.

This is a quick way of recovering from a command that is still being executed.

Pressing the `` key sends the `INTERRUPT` signal. (Some systems use the `<Ctrl>C` key for this purpose.)

Some programs deliberately ignore the key, such as the **vi** editor, the shells which process your commands, and the UNIX window (**scoterm**). These programs understand the keystroke as an instruction. For information about shells, see "Changing your shell" (page 108).

Ch

Fi

Ide

Vie

Specifying directory names

The UNIX system stores files in *directories*. A directory can contain files or other directories. Each user has a *home* directory. You can keep your personal files in your home directory, or create subdirectories within it to store files relating to certain categories. For example, your personal directory structure might look like this:

<code>./</code>	the home directory; contains ...
<code>./profile</code>	system file
<code>./login</code>	system file
<code>./mail</code>	subdirectory containing mail folders
<code>./mail/charless</code>	a mail folder
<code>./mail/charless/</code>	another mail folder
<code>./work</code>	subdirectory containing work files
<code>./work/proj1</code>	a file called <i>proj1</i> .
<code>./work/proj2</code>	a file called <i>proj2</i> .

`/u/charless` is the home directory. It is located in the `/u` filesystem. Within `/u/charless` there are some system files (`.profile` and `.login`) and some directories (`mail` and `work`). Each of the directories contains files partitioned according to their purpose; this makes it easier to keep track of a large collection of files.

NOTE This example is simplified and is intended to show the structure of a home directory, not an actual listing provided by a program such as `ls`.

The filesystem structure resembles an upside-down tree; each branch is a directory or subdirectory, and each leaf is a file. The main stem is known as the *root* directory. Each directory and file can be identified by a unique path in the tree.

To specify the path to a directory or file that is not in the current directory, you must give either an *absolute* or a *relative* path.

An *absolute* path lists all the directories and subdirectories you must enter to reach the target file, starting at the root (`/`) directory. For example:

```
/usr/fred/work/target
```

refers to the `usr` directory within the *root* directory, which contains the subdirectory `fred`, which contains another subdirectory, `work`, which contains the file or directory called `target` for which you are looking.

Build a *relative* path by specifying all the directories and subdirectories you must enter to reach the target file, starting from your current position in the filesystem. (To find out where you are, see "Identifying the current directory" (page 75).)

For example, supposing you are in */usr/me*, and want to specify a path to */usr/fred/work/target*. You can get there by the relative path:

```
../fred/work/target
```

The “..” symbol in your current directory represents the *parent* of the current directory, or the directory which is one level closer to the root directory; for example, *fred* is the parent of *work*.

You may also see a “.” symbol in a listing of your current directory. This refers to the current directory itself. For example, to refer explicitly to a file called *filename* in your current directory, you could type *./filename* or *filename*.

Finding a file

To check if a file is in the current directory, type **ls myfile** and press <Enter>. If it is present, the file name is displayed.

You can search for more than one file in a directory, but the files should have similar names for this to be practical. For example, supposing the current directory contains *my*, *myfile1*, *myfile2*, *myfile3*, *myfile4* and *myfile10*. To find all files that have names starting with *my*, type **ls my*** and press <Enter>.

The names of all files in the current directory that begin with *my* are listed. When you append an asterisk (*) wildcard character to the partially defined filename in the command, the system expands this to match filenames in this directory that start with *my* and are followed by zero or more characters. The asterisk wildcard “matches” any sequence of characters, including none at all, so *my** matches *my* as well as *myfile*.

To locate files with only one additional variable character, use the “?” wildcard character. For example, type **ls myfile?** and press <Enter>.

The system displays *myfile1*, *myfile2*, *myfile3* and *myfile4*, but not *myfile10*.

Typing **ls myfile??** and pressing <Enter> results in the filename *myfile10* being displayed.

To locate files with a range of characters in their names, enter (for example):

```
ls myfile[1-5][1-5]
```

This will list all the files starting with *myfile*, followed by two digits in the range one to five. (You can also specify a range of letters, for example [A-Z] or [a-z], and you can create sets of characters for which to search. For example [A-C1-90] matches the capital letters A, B, C or any digit.)

Chapter 9

Finding your way around

Files are stored in *directories*. A directory can contain files or other directories. Collectively, all directories and the files they contain are called a filesystem. The UNIX operating system provides a variety of tools for navigating the filesystem from the command line.

Identifying the current directory

To identify your current directory, type `pwd` and press `<Enter>`.

The current directory is the one you are currently working in. `pwd` stands for *print working directory*. The word *print* is used instead of *display* because the UNIX system was developed in the days of teletype terminals, when all output was printed.

Viewing the contents of a directory

To view the current directory's contents, type `ls` (short for *list files*) and press `<Enter>`.

`ls` lists the files and directories in the current directory.

To view the contents of a different directory, enter:

`ls directory`

where *directory* is the name of the directory whose contents are to be listed. See "Specifying directory names" (page 77) for more about referring to a directory other than the current one.

To modify the format or kind of information displayed, the following related commands can be used:

- l give a detailed listing of all filenames and their attributes
- lc give a listing of all files, split into columns
- lf give a listing of all files, split into columns, with directories followed by a slash (/) and runnable programs followed by an asterisk (*)
- lr list the contents of the current directory. If it contains any subdirectories, list their contents afterwards. If they contain subdirectories, continues listing their contents indefinitely. (The "r" is for *recursive*; for more about directories, see "Specifying directory names" (page 77).)
- lx give a listing of all files, ordered in rows

See the ls manual page for more about these commands.

Viewing the contents of large directories

The contents of directories that contain many files will scroll past faster than you can read them. To view the list one page at a time (where a "page" means as much text as you can put in the UNIX window at any time), enter:

`ls | more`

After each page `more` will display `more`; to see the next page, press `<Space>`.

In this command, `ls` is piped (`|`) to the `more` command. See "Running a sequence of commands" (page 73) for more information on using pipes.

Locating files outside your current directory

To locate a file that is *not* in your current directory, for example, *myfile*, enter:

```
find / -name myfile -print
```

`find` searches the specified directory and all directories branching from it. The direction of the search can be from the root (/) of the filesystem structure, as in the example above, or relative to your current position in the filesystem (or wherever you specify).

`-name` and `-print` are options to the `find` command for specifying the name of the file to search for and printing (displaying) the names of all matching files on the screen.

Changing directories

To change directories to a subdirectory of the current directory, enter:

```
cd mydir
```

In the example above, *mydir* is called a relative pathname because directions to it are given relative to the current directory. You may also use absolute pathnames with `cd`. Absolute pathnames specify directions to a directory from the top-level (root) directory. See "Specifying directory names" (page 77) for instructions on using pathnames.

To return to the previous directory from *mydir* using the relative pathname method, enter:

```
cd ..
```

This takes you back to the parent directory.

Creating directories

To create a directory called *newdir*, enter:

```
mkdir newdir
```

You may use either absolute or relative pathnames. The example above creates a subdirectory called *newdir* in the current directory.

Directories are created with access permissions that can be modified by the owner. For more information on permissions, see "Changing access permissions" (page 84).

To create a directory, you need to have write permissions for the directory within which you wish to create it, and your new directory name must be unique in this directory.

Removing directories

To remove a directory called *olddir*, enter:

```
rmdir olddir
```

You must not remove a directory with **rmdir** unless it is empty (contains no files or subdirectories) and you have write permission. See "Removing files" (page 92) and "Changing access permissions" (page 84).

You can not remove a directory if you are in it. To remove the current directory, first change to another directory.

tory
t be

s no
files"

irrec-

Chapter 10

Working with files

A file is the basic unit in which the UNIX system stores information. While you may see references to sub-units, such as records and fields, the file is the smallest block of information that is stored by name and recognized by the UNIX system.

Although most of your work with files is done with Desktop tools (such as Edit), the UNIX system provides a rich assortment of tools for manipulating files from the command line. Tools are available to work with files (moving, copying, deleting and changing them) and to work within files (searching for information, editing, and sorting).

Moving and copying files

When you *move* a file, you are placing it in another directory (or under another name in the same directory). When you *copy* a file, you are creating a duplicate, which occupies additional space in the filesystem. When you create a new *link* to a file, you are giving the file an additional name: the file is stored in only one space, but the linked names may appear in several directories.

Creating files

You create a file whenever you make a copy of a file, edit a new file, or cause the output of a command to be directed to a file that does not yet exist. See "Specifying command input and output" (page 72).

Although you will usually use Desktop editors to create or change files, you can edit files from the UNIX command line with the *vi* editor. See "Using *vi*" (page 109).

Moving files

To move a file to another directory, enter:

```
mv filename destination
```

where *filename* is the name of the file you want to move (preceded by its path if it is in a directory other than the current one), and *destination* is the path to the directory where you want to put it. The file disappears from its original directory and reappears in the destination directory.

To change the name of a file within the current directory, enter:

```
mv old new
```

where *old* is the file's current name, and *new* is the file's new name.

You can combine these techniques to move a file to a different directory *and* give it a new name at the same time. For example:

```
mv chapter.1 /u/workgroup/finished.chapter.1
```

moves *chapter.1* to */u/workgroup* and renames it *finished.chapter.1* at the same time. You can only use this technique to move files if you have write permission to them.

Copying files

To make a copy of a file, enter:

```
cp old new
```

where *old* is the name (preceded by its path if it is in a directory other than the current one) of the file you want to copy, and *new* is the name for that copy.

new does not have to be in the same directory as *old*. See "Moving files" (this page).

Combining files

You can combine two files, end to end, using the `cat` command. `cat` simply copies its input to its output. You use it like this:

```
cat file1 file2 > file3
```

where *file1* and *file2* are files to read (the input) and *file3* is the name of the file to create (`cat`'s output). `cat` reads *file1* writing a copy of it to *file3*, then reads *file2*, appending it to *file3*.

Linking files

To create a *link* to a file, enter:

```
ln file1 file2
```

A *link* is a new file name (or "link") that refers to a file that already exists; in effect, the file has two (or more) names. The names, or *links*, do not need to be in the same directory, or have the same owner.

You can use links as a shortcut to edit a file in another directory, by creating a link to the file and keeping the link in your home directory. Then, whenever you want to edit the file, instead of changing to the other directory, you can just edit the link.

You can also use links as a shortcut when changing directories. Create a link to a directory using the *-s* (symbolic) option, and *cd* into the link. For example, suppose you work in */u/workgroup/tasks/project/01* and your home directory is */u/me*. Your normal command to work on a file is

```
cd /u/workgroup/tasks/project/01
```

but you can create a link:

```
ln -s /u/workgroup/tasks/project/01 01
```

that creates a link in your current directory. Then you can move around like this:

```
$ pwd
/u/me
$ cd 01
$ pwd
/u/workgroup/tasks/project/01
```

The *-s* option means that the link is *symbolic*; it points to a file on a different filesystem, or to a different type of file (such as a directory).

See "Removing files" (page 92) for instructions for removing a link.

Changing files

You can change files by altering their names, changing their attributes (permissions or properties), and working on their contents (for example, by sorting them). These operations can be performed on the Desktop, but can also be accomplished from the UNIX command line.

Renaming files

To rename a file, use the **mv** command. For example, the command:

```
mv file1 file2
```

renames *file1* to *file2*.

You can use this technique to rename directories if you have write permission to them.

Changing access permissions

If you cannot look at a file, you probably do not have *read* permission for that file.

To find out if you have read permission, enter:

```
l filename
```

If the second character position in the ten character field at the left of the listing is "r", the owner of the file has read permission on the file. The login name of the file's owner is listed in the second field in the listing. If you are the file's owner but the left (owner) read permission is not set, you can give yourself read permission like this:

```
chmod u+r filename
```

This **chmod** command modifies the permissions on *filename* so that the owner (denoted by "u" for user) is given read permission (denoted by the "r").

Only the owner of a file can use **chmod** to alter the permissions on that file.

To give yourself permission to write to or *execute* (run) a file, use either the **chmod u+w** or the **chmod u+x** version of the command.

You can give members of your work group permission to read, write and/or execute the file using the **g+r**, **g+w**, or **g+x** versions of the command, if you own the file.

You can make a file publicly accessible using the **o+** form of the command ("o" is for *others*, meaning all other users of the system.)

To revoke permission to read, write or execute a file for a type of user, use a "-" instead of a "+"; for example, to revoke read permission for other users, use:

```
chmod o-r filename
```

Sorting files

To sort a file containing lines of text or numerical data in a variety of ways, use `sort`. For example, to sort a file (called *filename*) containing lines of text into dictionary order, regardless of upper- or lowercase:

```
sort -df < filename > sorted
```

The `-d` option specifies that "dictionary" order is to be used for the sorting process. The `-f` option means that lowercase letters are folded into uppercase (capital letters) for purposes of comparison. The file called *sorted* contains the result of the sort operation on *filename*.

To combine two files into one new file, the contents of which are sorted, enter:

```
sort -u file1 file2 > file3
```

This creates a file called *file3*, containing the sorted, merged contents of *file1* and *file2*. (`sort` sorts the files as it merges them.) The `-u` option tells `sort` to make sure that each line in *file3* is unique; that is, if both *file1* and *file2* contain an identical line, only one copy of the line will be written to *file3*.

The `-r` option reverses the order of the sort. Use the `-n` option when sorting lists of numbers, so non-numeric characters in the numbers (minus signs, decimal points, and leading spaces, for example) are not sorted incorrectly. The `-M` option makes `sort` assume that the first three characters of the field being sorted are months (like JAN, FEB, MAR, and so on) and sort them into date order.

You can make `sort` pick any portion of a line on which to base its comparisons. For example, to sort a list of names followed by months on the basis of the month:

```
sort -M +1 < filename > sorted
```

will sort a file containing lines like

```
martin FEB
angela DEC
judith JAN
```

into

```
judith JAN
martin FEB
angela DEC
```

The `+1` option tells `sort` to make comparisons between lines on the basis of the second field of each line. (A field is a sequence of characters separated by spaces or tabs; `sort` counts fields starting from zero.) So the "month" abbreviation on each line of the file is used as the basis for the sort operation above.

If you have a file where data records are made up of fields separated by some special character like a colon ":" (called a *separator*), you can tell `sort` to use a different separator by using the `-tseparator` command.

For example, `-t:` causes `sort` to split lines into fields separated by colons.

Looking inside files

The UNIX system does not normally distinguish between types of file. There are many different types of files in the filesystem, some of which you can work on and some of which you should avoid. For example, you can edit text files with the `vi` editor, and you can also read in and edit program files with it, although this is not a useful thing to do. It is more efficient to use the specific UNIX tools for identifying the type of information files contain.

Identifying file type

You can find out what type of information a file contains using:

`file filename`

`file` looks at the contents of a file and tries to determine what type of information the file contains. `file` can tell whether it is an executable program, contains data used by a program, or is text in English or another language.

It is a good idea to use `file` before examining the contents of a file as described below; if you try to examine a binary (or executable) file you may render your display unreadable, because binary files often contain characters that are interpreted as control codes by the terminal.

Previewing files

To look at the first or last lines of a text file, use `head` or `tail`. For example:

`head filename`

displays the first ten lines of `filename`, while:

`tail filename`

displays the last ten lines of `filename`.

If you use a numerical option, for example `-20`, `head` or `tail` will display 20 lines instead of 10.

Viewing very short files

You can look at the contents of a short file, using:

```
cat filename
```

`cat` concatenates its input to its output. This means that `cat` sends *filename* to your window, for you to read. (As your output is usually your window, `cat` sends its input there unless you tell it to use an output file).

You can `cat` more than one file at a time; the files are listed one after the other.

Hint: if you do not know what is in a file you want to `cat`, try using:

```
cat -v filename
```

This will cause any unprintable characters in *filename* to be displayed in a manner that will not corrupt your window.

See "Combining files" (page 82) for more information about `cat`, input, and output.

Viewing longer files

To look at the contents of a file that is too big to fit in a single window, enter:

```
more filename
```

`more` displays files one page (window) at a time. After filling the window with text, `more` will display the prompt `more`. To see more, press (Space).

While `more` is running, you can search for text in a file by entering a "/" followed by some text to find; for example:

```
/something
```

will make `more` search forward until it comes to the next occurrence of the word "something".

If you accidentally read past a piece of text you want to look at, press `b` to jump back a page.

To quit `more`, type `q`.

There are a variety of other commands which `more` recognizes. If you want a list of them, press `h` for help when you are viewing a file using `more`.

Searching files

If you want to find which file contains some specific word or words; there are UNIX tools to help you. You can search files rapidly to locate a known piece of text, or you can search files more slowly to locate a piece of text when you are unsure of the spelling.

Searching for text

To search a file for a specific piece of text, enter:

```
fgrep text file1 [file2 file3 ... ]
```

fgrep ("fast grep;" see "Using wildcards" (this page) for more about **grep**) searches all the files in *file1*, *file2* and so on for the specified *text*, and reports any matches.

To search for a text containing spaces or tab characters, enclose the text in quote marks.

To search for a text containing quote marks, put a backslash immediately in front of each quote character within the text.

If you are not sure whether the text is uppercase, capitalized, or all lowercase letters, type **fgrep -y**. **fgrep** will then ignore the case, and report all matches.

To see all lines in a file that do *not* contain the text, type **fgrep -v**.

Using wildcards

You can search files for text when you are unsure of the spelling of the text by using wildcards and **grep**. (**grep** is an acronym for "global regular expression print;" a "regular expression" is a complex wildcard.) For example, to search a file for the word "center" when you are not sure whether it is spelled the American way ("center") or the British way ("centre"), or even if it is present in another form ("central" or "centrally"):

```
grep 'cent[er]' filename
```

The "[er]" is a *set* containing the characters "e" and "r". This means that **grep** will search for the text "cent" followed by either an "e" or an "r". (The single quotes are needed because otherwise "[er]" will be interpreted as a wildcard by the shell, before the command is passed to **grep**.)

If you know that you want either "centre" or "center" but not "centrally" or some other combination, you can search for:

```
grep 'cent[er][er]\ ' filename
```

The "\ " means "followed by a literal space character." (Remember, you need to enclose the search text in single quotes because it contains a space.) So 'cent[er][er]\ ' will match "centre " or "center " but not "centerpoint " because the two characters are not followed immediately by a space.

You can put more than two characters in a set. For example, you can search for any character in the set [ABCDEabcde]. To save typing, you can enter this set as [A-Ea-e], where the "-" indicates that your wildcard is a *range* of characters. Alternatively, instead of using the "[...]" set notation, you can search for the special set "." (that matches any character except a newline).

To search for a sequence of characters in a set that repeats an indefinite number of times, you can use the "*" symbol after the set of characters for which you are searching. "*" matches zero or more occurrences of the preceding wildcard. For example:

```
grep 'cent.*' filename
```

searches *filename* for the letters "cent" followed by a sequence of zero or more characters matched by "." (the set matching any single character). Because "." matches everything except a newline, this wildcard will match "center", "center", "central", "accentuate" and "centipede".

You can also match anything that is *not* part of a set. If you want to find every line that contains a "cent" unless it is in "centipede" use the following command:

```
grep 'cent[~i]' filename
```

The "~" at the beginning of the set means "match any character *except* a member of this set."

You can search for text at the beginning or end of a line, using the "^" or "\$" symbols respectively. For example:

```
grep '^begin' filename
```

searches for all lines beginning with the string "begin" while:

```
grep 'end$' filename
```

matches all lines with the string "end" as a terminator.

e are
ce of
u are

grep)
ports

xt in

ly

rcase
es.

xt by
ssion
rch a
d the
esent

grep
ngle
lcard

Finding out how large a file is

You can find out how large a file is by using the `wc` command:

```
$ wc file
  684   4380  30675  2110
```

`wc` counts the number of lines, words, and characters in a file (in that order). You can use `wc` to get specific totals in any order using the options `-l`, `-w`, or `-c` to stand for lines, words or characters respectively. For example, to see the number of characters and lines in a file (in that order):

```
$ wc -cl file
 2675  684  2110
```

You can also give `wc` a list of files to count. For example:

```
$ wc chap1 chap3
 105   476   3244  chap1
 675   4620  31269  chap3
 780   4545  28113  total
```

Extracting fields

If you have a file containing columns of data in textual form, you can extract information from it using a variety of tools. For example, suppose you have a file called *blackbook* containing names, extension phone numbers, login names and dates, in a format like this:

```
Michael Stand:571:mike: JAN-1-91
Sue Penny:284:sup: FEB-6-89
Joshua Ford:454:jo: apr:04-78:88
Eric Adkerson:501:123: AUG-15-91
...
```

To see Sue Penny's record, use the following command:

```
$ grep Sue blackbook
Sue Penny:284:sup: FEB-6-89
```

This is hard to read. To see only Sue's extension number (the second field), you can use the `cut` command:

```
$ grep Sue blackbook | cut -f2 -d:
284
```

`cut` extracts individual fields from a file containing records on separate lines. The `-f2` option tells `cut` to extract only the second field of each record; the `-d:` option means that fields are delimited with a colon (:) instead of tabs.

The `|` is called a *pipe*; it tells `grep` to send its output to another program (in this case, to `cut`) instead of the window (or "standard output"). See "Combining files" (page 82) for information about pipes, input and output.

To see a list of all the people in your file, followed by their logins, you do not need to use `grep`. Just use `cut`:

```
cut -f1,3 -d: blackbook
```

The `-f1,3` option tells `cut` to extract the first and third fields in each record:

```
Michael Stand:mikes
Gae Penny:puep
Joshua Ford:joshp
Liz Addams:liah
```

To alphabetize your list, you can sort it like this:

```
$ cut -f1,3 -d: blackbook | sort -df
Joshua Ford:joshp
Liz Addams:liah
Michael Stand:mikes
Gae Penny:puep
```

Printing a file

To print a file, enter:

```
lp filename
```

`lp` responds with

```
request id is laserwriter-635 (1 111-)
```

This command sends *filename* to the print queue. (`lp` is short for “line printer”.) The “request id” line means that the file will be printed on the printer named “laserwriter”, and has the job number laserwriter-635.

A print queue is a queue of files waiting to be printed on a specific printer. Because a UNIX system may have many users, any or all of whom may be printing files, your file might not be printed at once. It goes onto the back of the queue. However, if it is the only file waiting to be printed, it will be processed at once. Otherwise, it has to wait for its turn.

If you know that several printers are connected to your system, and you want to send a file to a printer that is not busy, you need to know the destination printer’s name. Use the `-d` option to `lp` to choose the destination; for example:

```
lp -d fast_printer filename
```

sends *filename* to the printer named *fast_printer*.

You can get a list of the printers available to you by using the `lpstat` (line printer status) command:

```
lpstat -s
```

der).
or -c
the

tra
ive a
imes

eld),

ines.
e -d:

n (i
bit.

To cancel a print request, if it has not yet printed, use the **cancel** command and the request-id you were given when you entered the **lp** command to print the file. For example:

```
cancel laserwriter-635
```

cancels the print job "laserwriter-635". If you do not remember the request-id, enter the **lpstat** command with no options to see a list of print jobs in the queue.

Removing files

It is necessary to remove files from time to time, to prevent the filesystem from filling up. However, you should be extremely careful about removing files. Although the Desktop environment lets you retrieve deleted files, once a file has been removed it is gone forever; there is no way to get back any information that you have lost. Therefore, you should use the **rm** (remove) command with care.

Removing ordinary files

To delete a file, use the **rm** command. For example:

```
rm -i filename
```

rm -i is *interactive*; when you see the question mark, you can either type "y" in which case it will destroy *filename*, or "n", in which case it will not.

It is a good idea to use the **-i** (interactive) option with **rm** because once you have removed a file, it is impossible to get it back again.

To remove several files, you can type **rm -i** and use wildcards to select them. (Always use the **-i** option with **rm *** unless you are absolutely certain that it is safe to delete everything in the current directory.)

Removing links

You can remove a link you no longer want, just like a normal file, with **rm**.

```
rm link
```

deletes a link called *link*, just like a normal file.

The file itself will not be deleted until the last link (or name) by which it is referred to is deleted.

Searching for lost files

If you have put a file somewhere and cannot remember where it is, you can use the `find` command to locate it:

```
find /-name filename -print
```

The `/` tells `find` to start searching in the *root* directory. `find` searches its starting directory, and all the subdirectories it can find, in order. If you know your file is in one of your own subdirectories you could tell `find` to start searching from `$HOME` instead of `/`. `$HOME` represents your home directory; see "Setting variables" (page 105) for details.

The `-name` option is followed by the name of the file for which you are looking. Every time `find` sees a file with this name, it will carry out the actions specified by the subsequent options.

The `-print` option tells `find` that the action to take when it finds `-name filename` is to print (display) its full path and name on your screen.

`find` can carry out other tasks besides showing a file's full path. For example, the command:

```
find /bin -name filename -exec l {} \;
```

causes `find` to execute `l` on any file it finds with the name `filename` under the directory `/bin`. (The `{}` in the `-exec` command stands for the name of the found file; the `\;` marks the end of the `exec` option.)

```
find /-name chap3 -print
```

If the command above results in a series of error messages like:

```
/usr/sbin: not found /usr/sbin/chap3
/usr/sbin: not found /usr/sbin/chap3
find: cannot chdir to /etc/crontab/packed/atp
find: cannot chdir to /etc/crontab/packed/atpproc
find: cannot chdir to /etc/crontab/packed/atpb9
...
```

you can ignore the errors by re-issuing the command as follows:

```
find /-name chap3 -print 2> /dev/null
```

The error messages are coming from `find`'s *standard error*, because `find` does not have permission to read these directories on your behalf. The fragment

```
2> /dev/null
```

tells `find` to send its error output (the *standard error*) to `/dev/null`. (`/dev/null` is a *device file* that suppresses any output stream you send to it.)

In general, if any program gives a series of error messages, you can stop them from cluttering up your window by adding "`2> /dev/null`" to the end of the command line, or you can add them to a log file by adding "`2>> error.file`" to your command. (Note that this will not work if you are using the C shell rather than the Bourne or Korn shells.)

Chapter 11

Copying files to disk and tape

Most of the time, you work with files in the filesystem, which is stored on your computer's hard disk. However, sometimes you may want to copy files to and from tapes or floppy disks; for example, you may want to give a copy to a user on a machine not connected to your own, or to store infrequently-used material on a tape (which is much cheaper than space on the filesystem), or to make a back-up copy of your work.

There are many ways of saving and retrieving files from floppy disk or tape. The simplest method is to use a *tar* archive, as discussed in "Creating a tar archive" (page 96). Other methods are more complex or are only available to the system administrator.

You can also copy files to and from DOS floppy disks relatively easily. To use a DOS floppy disk, you must format the disk for DOS (page 99) if it is not already formatted, then use the DOS commands described in "Using DOS utilities" (page 99) to copy files between the UNIX filesystem and DOS floppy disks.

Formatting floppy disks

You must format a floppy disk before you can use it. To format a disk, ensure that it is in the appropriate drive and then enter:

```
format  
or  
format drive
```

If you do not specify the device, the default drive will be used.

drive is the *device file* the UNIX system uses to communicate with the type of disk you are creating. (The UNIX system sees all pieces of equipment attached to the computer as files; it communicates with them by reading from and writing to a special *device file* stored in */dev*.) There is a different device file for each different format of floppy disk.

You determine the name of the device file to use as follows:

1. All floppy disk devices are located in */dev* and begin with *rfd* (the "r" is short for *raw*, because the UNIX system has to access the disk directly).
2. If your computer has only one floppy disk drive, follow this with a number "0". If your computer has two or more drives, you can follow it with a "0", "1", or higher number (depending on whether you want to format a disk in the first, second or subsequent drive).
3. Follow this digit with the number of tracks per inch on the disk; for example, 135 if it is a high-density 3.5-inch floppy. (The number of tracks per inch, or "tpi", should be indicated on the disk or its case.)
4. Follow this number with either "ds" if the disk is double-sided, or "ss" if it is single-sided.
5. Finally, finish the device name by adding the number of sectors per track; 9 if it is a low-density 5.25-inch or 3.5-inch floppy, 15 if it is a high-density 5.25-inch floppy, or 18 if it is a high-density 3.5-inch floppy.

For example, to format a disk in the second floppy disk drive that is to be a high-density 3.5-inch double-sided disk, enter `format /dev/rfd1135ds18`.

`format` will prompt you to insert the floppy and press (Enter) to continue.

Note that it takes time to format a floppy disk — typically a minute or so (although this may vary).

Creating a tar archive

A `tar` archive can be thought of as a special file that contains other files and their associated directory information. To create a `tar` (tape archive) file on a floppy disk, enter:

`tar cvf device filename`

where *device* is the device file corresponding to the floppy disk, and *filename* is the name of the archive. (See "Formatting floppy disks" (page 95) for information about device files.)

List

Ext

tar lets you use an abbreviation to specify the device to use. To see the list of available types, enter:

```

% tar
usage: tar [-C directory] [-f device] [-F filename] [-b blocksize] [-t tapesize] files...

```

Key	Device	Block	Size(K)	Tape
0	/dev/rfd048ds9	18	360	No
1	/dev/rfd148ds9	18	360	No
2	/dev/rfd096ds15	10	1200	No
3	/dev/rfd196ds15	10	1200	No
4	/dev/rfd096ds9	18	720	No
5	/dev/rfd196ds9	18	720	No
6	/dev/rfd0135ds18	18	1440	No
7	/dev/rfd1135ds18	18	1440	No
8	/dev/rct0	20	0	Yes
9	/dev/rctmini	20	0	Yes

Using this list, you can select the size of disk you want to use. For example, to create a tar file on a 720K floppy disk in the second floppy disk drive, just use the command **tar cv5 filename** (where *filename* is a list of files to create in a tar archive on */dev/rfd196ds9*, separated by spaces).

If you do not specify a device, the tar archive will be created on the default device (specified in */etc/default/tar*).

If you specify a *filename* instead of a device name with the **-f** option, tar will create the archive in your current directory (or the path indicated by *filename*; it looks like a special type of file, called a *tarfile*). The tar archive will be created and copies of all the files will be stored in it.

Listing tar archives

Type **tar tvf device** and press (Enter). (*device* is either the name of the device containing the floppy or tape where the archive is stored, or the name of the file containing the archive.)

Extracting tar archives

To extract files from a tar archive, type **tar xvf device filename** and press (Enter). tar looks inside the archive on *device* (or the file of that name) and extracts any file it sees which matches *filename*.

For information about other options to tar, see the tar manual page.

Understanding magnetic tape

You can copy files to and from tape devices using `tar`, in the same way as you deal with floppy disks. However, there are a number of differences between tape and floppy disk systems. Notably, although magnetic tapes can store far more data than a floppy disk, they can only provide *serial access* to the information. That is, when reading or writing a tape you must start at the beginning and read through each file stored on it in order until you get to the end — you generally cannot jump around or skip files. Consequently tapes cannot be used as filesystems.

To copy files to and from a tape device you should use `tar`, with the appropriate device file (from the list below). You may also need to use the `tape` command to control the tape drive directly; see "Rewinding and erasing tapes" (this page).

There are several different types of tape which may be available. The commonest are:

- QIC-02 A full-sized quarter-inch tape cartridge, the first QIC-02 drive uses the `/dev/rct0` device file.
- QIC-40/QIC-80 These smaller mini-cartridge units related to the QIC-02 format are accessed through the `/dev/ft0` device file.
- mini-cartridge Mini-cartridge tape drives are linked to the floppy disk drive controller and differ significantly from the QIC family of tape drives. Notably, you must format mini-cartridge tapes before using them. They are accessed via the `/dev/rctmini` device.
- SCSI SCSI tape drives are controlled by a SCSI controller, like SCSI hard disks. They are accessed via the devices named `/dev/Stp0` to `/dev/Stp3` (or `/dev/rStp0` to `/dev/rStp3` for raw access).

For further information see the chapter on tape drives and controllers in the *Hardware Configuration Guide*.

Rewinding and erasing tapes

To rewind or erase a tape, you should use the `tape` command.

To rewind a tape, enter:

```
tape rewind
```

It is a good idea to rewind the tape to the beginning after every use, or after encountering an error.

To erase a tape, enter:

tape erase

It is not necessary to erase a tape before reusing it. However, you may want to erase a tape for security reasons.

You should retension any tapes that you use regularly, or that have been in storage and from which you now wish to read. This takes up any slack in the cartridge and reduces the likelihood of errors. The command to retension a cartridge is **tape reten**.

In addition, you should write-protect your tapes to prevent accidental erasure or overwriting. This is done by turning the slot on the cartridge to the SAFE position; turn it the other way when you intend to write over or erase the tape.

Formatting a DOS floppy disk

The UNIX system provides special tools for manipulating floppy disks that are compatible with DOS.

To format a floppy disk for use with DOS, enter:

dosformat device

where *device* is a special file, as explained in "Formatting floppy disks" (page 95).

A DOS format disk cannot be used with **tar**; to store files on it you must use the special DOS utilities described in "Using DOS utilities."

Using DOS utilities

Several special UNIX commands are provided for manipulating DOS disks (not to be confused with the actual DOS commands provided by Open Desktop DOS Services). They are as follows:

dosls drive

provides an ls style listing of the files stored on *drive*, where *drive* is either a UNIX-style device file or the DOS drive name (A: or B:)

dosdir drive

similar to **dosls**, but provides a directory listing after the style of the DOS program **dir**

doscp filename destination

copies the file *filename* to the specified *destination*. You can copy files to a subdirectory on the DOS disk, if you specify the pathname (for example, *A:\MYDIR\MYFILE.TXT*). You can use **doscp** to copy files to a DOS disk from a UNIX system, or to a UNIX system from a DOS disk.

Note that **doscp** does not recognize wildcards; if you want to copy more than one file using wildcards, you should enter the following:

```
for file in wildcard
do
  doscp $file destination
done
```

where *wildcard* is used to identify the files you want to copy, and *destination* is where you want to copy them.

dosrm filename

deletes the named file. Note that you can give a pathname, if the file is in a subdirectory on the DOS disk.

DOS filenames are different from UNIX filenames. The following rules apply:

- case All DOS filenames are uppercase. UNIX files are converted to uppercase when they are copied to DOS, but DOS files remain in uppercase when they are copied to a UNIX system. (DOS is not case sensitive).
- paths Paths are separated by a backslash (\), rather than a slash (/).
- length DOS file names are limited to eight characters (called the file name) followed by a period, followed by three characters (called the extension). UNIX files with names which are too long lose the trailing letters.
- links DOS does not recognize links. If you use **doscp** to copy a link to a DOS disk, a complete copy of the file is made. So, if you have two links to the same file called *file1* and *file2*, and copy them both to the same DOS disk, the result will be two identical copies of the file, named *FILE1* and *FILE2*.

You should write-protect your back-up floppy disks to prevent accidental erasure or overwriting. On 5.25-inch floppies, cover the square notch on the side of the disk with the supplied write-protect sticker. On 3.5-inch floppies, slide the write-protect tab closed.

If you give this script to a friend, it will do the same for that user — searching only the files below that person's home directory — because the value of `$HOME` is unique to each user.

Changing your password

To change your password from the UNIX command-line, enter `passwd`.

`passwd` will ask you to type in your old password before you select a new one. If you choose to select a password of your own, `passwd` will ask you to enter it twice, and will make sure that it is not a word that is easy to guess. If you do not choose a password, `passwd` will generate a suitable password for you.

You should take care not to write your password down anywhere, not to tell it to anyone, and to choose one that is difficult to guess (avoid names, places, or telephone numbers). For further information about choosing a password, please see the Open Desktop *System Administrator's Guide* or the `passwd` manual page.

Note that if you forget your password you will not be able to log on again. You will have to ask your system administrator for help.

Setting up environment variables

It is possible to set environment variables automatically when you log on. Whenever you log on, your shell looks for a special login command file (or startup file) called either `.profile` or `.login` and executes any commands it finds in it.

To find your login command file, enter:

```
ls -a
```

There may be more than one file in the listing. If you are using the C shell, you need to edit `.login`. If you use the Korn shell or the Bourne shell you must edit `.profile`.

To add a new variable to your startup file, simply edit the file and insert two lines like:

```
my_variable=Seven
EXPORT my_variable
```

(Note that this does not apply to the C shell.) This will set the value of `my_variable` to "Seven" and *export* it so that sub-shells can make use of it. (An unexported variable is only available to the shell within which it is defined.)

A sub-shell is a shell run from within another shell. For example, when you execute another command from within the vi editor, it is running in a sub-shell. For more information on shells, see "Changing your shell" (this page).

Changing your shell

If you want to change your default shell (the one you use when you start work), you should ask your system administrator. Information on how to change the default shell is provided in the *System Administrator's Guide*.

The shell is the program that the UNIX system uses to communicate with you, the user. The shell reads your instructions and carries them out, locating and running programs and interpreting scripts. (It is called the shell because it puts a shell around the core of the UNIX system, making it easier to work with.)

The following shells are available:

- | | |
|---------------|---|
| Korn shell | The shell of choice, it provides facilities for recalling and editing commands you have already typed in, and for controlling background programs. |
| Bourne shell. | The original UNIX shell, it predates the Korn shell and offers fewer facilities. |
| C shell | This shell has a C-like syntax with basic command recall facilities. It is incompatible with the Korn and Bourne shells. |
| SCO shell | This is a menu based shell, designed to help inexperienced users master the complexities of the UNIX system without recourse to the other, command-line shells. |

Remember, if you request a change of shell, you must move any environment variables you have set up into the appropriate file in order for them to be set when you start a UNIX session. The files to check are *.cshrc* and *.login* for the C shell, *.profile* for the Bourne shell, or *.kshrc* and *.profile* for the Korn shell.

See "Where to find more information" (page xviii) for sources of more information about shells.

If you want to write a script where *all* the parameters, however many there are, are passed to a program, use the variable `$*`. For example, a script containing the following:

```
file $*
```

will run `file` on every item specified on the command line, whereas:

```
file $2 $3
```

will only run `file` on the second and third parameters.

Setting variables

A variable is a label the UNIX system uses to refer to some variable quantity it needs to track. Each variable has a *name* and a *value*, that is stored in the variable.

UNIX variables are known collectively as the *environment*. Many programs use variables to store information temporarily.

To find out what variables are currently set, enter:

```
env
```

You will see a long list of information, looking something like this:

```
LANG=en_US.UTF-8
PATH=/usr/sbin:/usr/bin:/usr/X11/bin:/usr/local/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin:/usr/sbin:/usr/bin:/usr/X11/bin/...
```

The name on the left of an equals sign is that of a variable; the information on the right is the value associated with the variable. For example, `LANG` is a variable name, and `english_us.ascii` is its value.

To set a new variable, enter:

```
myvariable=value
```

where *value* is whatever you wish `myvariable` to equal. (If you are using the C shell instead of the Bourne or Korn shells, you will need to enter `setenv myvariable value` instead.)

If you want to make `myvariable` accessible to all programs you run, follow it with the command `export myvariable`. (This is not necessary in the C shell.)

For example, suppose you want to make the shell load and run programs stored in a directory called `/u/myprogs/bin`. When you enter the name of a program, the shell looks for it in those directories that are listed, separated by colons, in the variable called `PATH`. To add `/u/myprogs/bin` to your `PATH` variable, enter:

```
PATH=$PATH:/u/myprogs/bin
```

This replaces the current value of `PATH` with itself, followed by `:/u/myprogs/bin`.

To make this available to all programs you run in the remainder of this UNIX session, enter:

```
export PATH
```

Removing variables

To remove a variable, enter:

```
unset variable
```

where *variable* is the name of the item to remove. (In the C shell, enter `setenv variable ""` to remove *variable*).

Referring to variables

You can refer to the contents of a variable within one of your scripts. To do so, insert the name of the variable, preceded by a "\$" symbol. The value of the variable will then be substituted. For example, to save your current directory in a variable, create a file containing the following:

```
current=`pwd`
```

This command assigns the output from running `pwd` to the variable `current` when you run it.

If you `cd` to another directory, you can return to this one by typing:

```
cd $current
```

from wherever you are.

For example, the variable `$HOME` contains the path to your home directory. Suppose you want to find files located in one of your subdirectories. You can use a script like this:

```
find $HOME -name $1 -print
```

Supposing the script is called *whereis*, and you want to find a file called *new.document*, you would enter `whereis new.document`. `whereis` will only search those directories that are located within your home directory.

Chapter 12

Controlling the work environment

The programs you run define your work environment. The UNIX system is extremely customizable; you can add and remove programs to make your work easier, you can change the priority of jobs (programs that are running), and you can set up the UNIX system so that specific programs run and their required variables are loaded whenever you start a new UNIX session.

Improving performance

The UNIX system is multitasking; many programs may be running simultaneously. You can make the UNIX system operate more efficiently by changing the priority the UNIX system assigns to individual programs, running long programs in the background, and removing programs that are not doing what you expect. But first, before you can do any of these tasks, you need to know what programs are running.

Determining which programs are running

To find out what programs you are running, enter **ps** (process status). This will display information about your current programs, in columns for PID (Process ID; see below), TTY (the terminal on which the command is running), TIME (elapsed time) and COMMAND (the name of the program).

To find out all the programs running on the system that you are authorized to see, enter **ps -ef | more**. This shows all the programs that are running, rather than just your own. It also provides information about the PID of the program's parent, the UID (identity) of its owner, and the current state of the program.

Among the programs that are running, you will see `ps` and `more`. When you entered `ps -ef | more`, you created two new *processes*. (A "process" is a program that is loaded and is running.)

Running programs in the background

You can run a non-interactive program "in the background" (so that while it executes, you can get on with something else) by using the "&" notation. For example, to run `sort` in the background, enter:

```
$ sort file > sorted &
2360
$
```

The number that appears before the prompt is the Process ID (PID) of the `sort` command. If you want to stop the process before it completes, you will need to use this number.

It is not appropriate to run interactive commands such as `vi` in this way.

Continuing programs after logging off

To run a background program that will continue after you log off, enter:

```
nohup program_name &
```

`nohup` means "no hang-up". A program started in this way will continue until it finishes and will not be aborted by your UNIX session's end.

For example, if you are about to print a very long file using the text formatter `nroff`, but need to log out in order to go and do something else, you can enter:

```
nohup nroff myfile > formatted
exit
```

`nroff` runs in the background and does not stop when you log off. Any error output from the program will be saved in a file called `nohup.out`.

Managing demanding background jobs

To reduce the demands a program makes on the UNIX system, use the `nice` command. For example:

```
nice -20 find / -name something -print > outfile &
```

`find` runs in the background, sending its output to `outfile`. By using `nice` with a value of "20" to run it, you make the UNIX system spend less time attending to `find` than to any other programs running at the same time.

`nice` needs a number to tell it how "nice" to be to other users. (A "nice" program is one that does not take over the system.) The number is between 1 and 20; a small value is less "nice" than a large value.

A demanding background process (such as a compiler or text formatter) can slow down the whole system. You can make life easier by reducing the amount of time the UNIX system spends on that program. It will take longer to run, but will not slow down your other tasks.

Stopping runaway processes

If you have started a process running in the background and need to halt it before it finishes, you can use the **kill** command. For example:

```
kill -15 2360
```

kill sends a *signal* to the target process. A signal is a special message with one of several pre-defined values. The first number (after the "-") identifies the signal to send; signal 15 is a command to terminate. The second number is the PID (process ID number) of the process to which to send the signal.

You can obtain the PID of a process using the **ps** command; see "Determining which programs are running" (page 101). If you know the name of the program you want to stop, you can use the following command to find its PID:

```
ps -u login_name | grep program_name | grep -v grep
```

where *login_name* is your login name (truncated to seven characters), and *program_name* is the name of the program to find. The PID is the number in the second column.

If **kill -15** fails to halt a program that is out of control, try **kill -9** instead. This is more effective, but does not give the program a chance to close any files it may be working on when it receives the signal.

You cannot kill processes belonging to another user or to the system (unless you are the root, or super-user).

Running command sequences

You can run a sequence of more than one command from a single command line. To send several commands, one after another, separate each of them with a semi-colon; for example:

```
ls > list; vi list; sort list
```

This command sequence creates a list of files in a file called *list*, runs the **vi** editor on the list, then sorts it. (Note that you cannot run **vi** on the data in a pipe.)

If you want to repeat this sequence of commands, you can write them in a file, then make the file execute as a command. This type of file is called a "script" or a "shell script."

For example, to put the preceding command sequence into a file called *myscript* that can be used as a command:

```
echo 'ls > list ; vi list ; sort list' > myscript
chmod +x myscript
```

Now, when you enter *myscript* (or the name you gave to the file), the commands will be executed one after another.

Note that any file of commands you create must have its attributes set to "executable" before you can run it by typing its name: Otherwise, you will see a message like this:

```
myscript: cannot execute
```

For information on attributes, see "Changing access permissions" (page 84).

As an alternative, you can create a short file containing a list of commands; for example:

```
ls > list
vi list
sort list
```

that will be carried out in the same order when the file is executed.

Running scripts with parameters

You can make a file of commands run with different parameters specified on the command line by using *parameter substitution*. This feature lets you run a script using different data files or options for the programs listed in it. For example, suppose you change your file to read like this:

```
ls > $1
vi $1
sort $1
```

\$1 refers to the *variable* called "1". This variable is replaced with the first parameter specified on the command line. So if you enter:

```
myscript myfile
```

the word "myfile" is substituted for all occurrences of *\$1* in *myscript* as it is executed.

If you want to pass more than one parameter, you can use the variables *\$1*, *\$2*, *\$3*, ... to represent the first, second, third (and subsequent) parameters specified on the command line.

Sett

Chapter 13

Editing files

The UNIX system includes several editors, each optimized for specific needs:

- vi** This visual editor is used for interactively writing and editing files of text. **vi** is the editor you will use most in the UNIX window. It resembles the Desktop editor to some extent, but provides no menus or help. (There is a variant of **vi** called **vedit** that is set up for novices, and a variant called **ex** that behaves like **ed**, below.)
- ed** The original UNIX line editor, **ed** is *line-oriented*; it can only edit a line at a time. **ed** is used within shell scripts, and when it is impossible to configure a terminal properly.
- sed** The stream editor, **sed** reads its input file, carries out a sequence of commands, and writes the result to its output file. **sed** cannot be used interactively.

The UNIX system also provides a variety of tools for formatting, spell-checking and processing text. See Chapter 10 (page 81) for information on some of the programs available.

Using vi

vi is the standard UNIX tool for editing text. It differs from the Desktop editors in that, instead of being controlled through menus, it is controlled exclusively by commands you type into it.

Because it is designed to read commands from a variety of terminals, **vi** has two *modes*: insertion mode and command mode. If you are new to **vi**, you may find this confusing at first. Try to remember this simple rule: you cannot issue commands when in insertion mode (except for the command to switch to command mode), and you cannot enter text while in command mode.

you
sub-
-).

start
w to

you,
; and
ise it
work

; and
co

offers

recall
turne

nced
thout

ment
re set
the C

infor-

Starting vi

To start editing a file, enter:

vi filename

If the file already exists, vi will read it in. If it does not exist, vi will create it.

When you start vi, you are in command mode. vi has two modes; command mode, and insertion mode. In command mode you can issue commands to vi and move around your document. In insertion mode, you can only enter text.

Stopping vi

To leave vi you must switch to command mode, if you are not already in it. You can enter command mode by pressing `<Esc>`. The terminal beeps or flashes at you if you are already in command mode and press `<Esc>`.

There are several ways to leave vi. Here are the most common:

- ZZ** Save the current file and exit. (Just type a capital "Z" twice.) This command will not work if the current file is write-only, or you are attempting to edit more than one file. (This command is equivalent to `:w :q` or `:wq`.)
- :w** Save the current file (`w` is short for write file). Do not exit. This command will fail if the file is write-only. You can save under a different name by adding a filename: for example, `:w newfile` saves the current file as *newfile*. (Note that the colon (`:`) tells vi to read everything you type until the next `<Enter>` as a single command.)
- :q** Quit vi. This command will fail if the file has changed since the last time you saved it. (If you really want to quit without saving, enter command mode and type `:q!`. This causes vi to quit without saving any changes you have made to the current file.)
- !:cmd** Execute the program *cmd*, then return to vi. The command `!sh` is *not* the same as exiting vi; vi is still running, and when you exit the shell (by typing `exit` or `<Ctrl>D`) you will return to vi.

What to do if you encounter trouble

First press `<Esc>` twice. If a command is in progress, the `<Esc>` key cancels it. If you are in insertion mode, the `<Esc>` key puts you back into command mode. If your terminal beeps or flashes when you press `<Esc>`, it means you are now in command mode.

If the UNIX window is unreadable, press `(Ctrl)L` in command mode. `vi` then redraws (refreshes) the window.

If you still cannot read the UNIX window, either your terminal is set up incorrectly or you are editing a non-text file. Type `:q!` to exit without saving the current file.

Entering text

To type text into a file, you must switch from command mode to insertion mode.

To enter insertion mode, press `i` (for insert).

To leave insertion mode, press `(Esc)`. The terminal will beep or flash if you press `(Esc)` again.

If you are not sure which mode you are in, press `(Esc)` until the terminal beeps or flashes. You will then be in command mode.

When you are in insertion mode, anything you type is entered into the document at a position immediately behind the cursor. If you make a typing mistake, you can use the `(Bksp)` key to backspace over the error. When you have finished inserting text, press `(Esc)` to return to command mode.

Moving around inside the file

You must be in command mode before you can move the cursor around the file. If you are not already in command mode, you can enter it by pressing `(Esc)`.

To move a single character width in any direction, use the arrow keys on your keyboard. (The keyboard keys `"h"`, `"j"`, `"k"`, and `"l"` also move the cursor.)

You can move around in various units:

- | | |
|-----------|---|
| word | To move forward or backward a word at a time, press the <code>"w"</code> (word forward) or <code>"b"</code> (backward) key. |
| start/end | To move to the start of a line or the end of a line, press the <code>"^"</code> (start) or <code>"\$"</code> (end) key. |
| sentence | To move forward or backward a sentence at a time, press the <code>"})"</code> (next sentence) or <code>"("</code> (previous sentence) key. |
| paragraph | To move forward or backward by a paragraph, press the <code>"{ "</code> (previous paragraph) or <code>" }"</code> (next paragraph) key. |

window To move forward or backward by a window full of text, press the **<Ctrl>F** (forward) or **<Ctrl>B** (backward) key or the **<PgDn>** or **<PrevPg>** key.

line number "G" (goto). G without a number takes you to the last line in the file. If you enter 1G you will go to the start of the file.

To see your current line number, press **<Ctrl>G**. A status line will appear at the bottom of the UNIX window, telling you the name of the file, whether it has been modified, your current line number, the number of lines in the file, and your position in the file as a percentage of the length of the file.

To make any of these commands repeat, enter a number (the number of times you wish the command to repeat), then the command. For example, to move forward five words, enter 5w.

Deleting text

You must be in command mode before you can delete or change text.

To delete text, use the **d** command followed by the unit of text to delete. Options are:

dl delete letter (or type **x** — a shortcut)
dw delete word
dd delete line

To delete several units of text at a time, enter the number of units to delete, followed by the appropriate command. For example, to delete five words, type 5dw.

To delete a range of lines, enter a command in the form:

:x,yd

where **x** and **y** are the first and last line numbers in the range to be deleted. For example, to delete lines seven through seventeen inclusive, use the command:

:7,17d

Replacing text

To replace a single letter with another letter, position the cursor over the letter and type the `r` command, followed by the replacement letter.

To replace an unlimited amount of text on the current line with new text, position the cursor over the first letter and type `R`. You are now in replace mode. This corresponds to insertion mode, but characters you type will replace the previously existing text. You can return to command mode by typing `<Esc>`.

Inserting text

To insert text at the start of a line, regardless of where the cursor is within the line, type the `I` command.

To add text to a line, type the `a` command. This puts you into insertion mode, but text is added after the current cursor position.

To add text to the end of a line, regardless of where the cursor is in the line, use the `A` command.

Modifying text

To change the case of text (from uppercase to lowercase or vice versa), use the `~` command. Place the cursor over the text to change and press `~` once for each character. (The cursor advances by one character each time you issue this command.)

To swap two characters, position the cursor over the left character and type `xp`. The two characters will be transposed.

To open up a line below the line the cursor is on, type the `o` command.

To open up a line above the line the cursor is on, type the `O` command.

To join together two lines, type the command `J`. The line below the cursor will be joined onto the end of the line the cursor is on.

To undo the last command, type the command `u`. The result of your last command will be undone.

To undo all changes to the current line since you last moved the cursor to it, type the command `U`.

Cutting and pasting text using buffers

A **buffer** is where vi temporarily stores text. vi has twenty-six buffers, named "a" through "z".

To copy a line of text into a buffer, type the command

`"buffer_nameyy`

(the yy command is short for yank). To copy several lines, precede the command with the number of lines you wish to copy; for example, to copy fifteen lines into buffer "a", type "a15yy.

To delete a line of text, saving it in a buffer, use the command `dd` instead of `yy`.

To paste the contents of a buffer into the text immediately above the cursor, type the command `"buffer_nameP`. For example, to paste the contents of buffer "g" into your file above the cursor, type `"gP`. The paste command `p` (lowercase "p") pastes the buffer in below the current line instead.

You can cut or copy a region of text of any size with the *marker* facility. Place a marker in the text at the beginning of the region you want to move, by typing `m` followed by a letter ("a" through "z"). This inserts an invisible marker at your current position. Now move to the end of the region. To *copy* the text between the cursor position and the mark into buffer "a", type `"ay`a`. To *move* the text between the marker and the current cursor position into buffer "a", type `"ad`a`.

The command is built up as follows. First, specify that you are going to use a buffer by typing `"a` (for buffer "a"; you can use any other buffer you like). Second, specify whether you are going to cut text ("`d`" for delete) or copy text ("`y`" for yank) into the buffer. Third, use the ``marker` command to cut or copy the text between the current cursor position and the named marker. (``marker` means "go to *marker*" and *marker* stands for the mark you placed in the text.

Searching for text

You can search for text if you are in command mode. To search forward, type `/text`, where *text* is the text you want to find. If you find it, you can repeat the search for the next occurrence of the text by typing `/"` or `"n`.

To search backward, type `?"` instead of `/"`.

vi searches for text using wildcards, as does **grep**. The following wildcards are used:

- .
- (period) matches any single character except a newline. For example, `/g..d` matches "good" but not "god".
- *
- (asterisk) matches one or more instances of the last character specified. For example, `/f*` matches any number of 'f's, while `/.*` matches any number of any character.
- ^
- (caret) matches the start of a line
- \$
- (dollar) matches the end of a line
- [..]
- matches a set of characters. Any of the characters within the square brackets will be recognized.
- \
- (backslash) takes away the special meaning of the character to the right of the backslash. For example, `.*` matches any number of any character, but `\.*` matches a single asterisk.

Substituting text

To substitute one sequence of characters for another on the current line, use the `:s/old/new/` command, where *old* is the sequence to find, and *new* is the sequence that replaces it.

To substitute all occurrences of a sequence of characters within a file, type:

```
:g/old/s/new/g
```

You can search for wildcards, but should replace them with a string of ordinary text. For example, to search for any word beginning with "cent" (such as center, centered, or central) and replace it with "middle" instead, type:

```
:g/cent.*\ /s/middle/g
```

Configuring vi

vi has a number of internal variables that can be configured with the `:set varname` command, where *varname* is the name of the variable to change.

To examine the state of vi's settings, go to command mode and type `:set all`. You will see a list of settings.

If a variable name starts with "no", it is not set (that is, not switched on). You can set it by typing `set varname`. If a variable name does not start with "no", and is not followed by a number, it is set. For example, if you want to make vi ignore wildcards, you must switch off the variable **magic**. To do this, type `set nomagic`. If the variable name is followed by a number, you can change its value by typing `set varname=value`, where *value* is the new setting you want it to have.

To make **vi** automatically begin a new line before you reach the right side of the UNIX window, type `:set wrapmargin=15`. This makes **vi** "wrap" the first word you begin to type that is less than fifteen characters from the right side of the window. (If you have used other word processors, this feature may be familiar to you as "word wrap.")

A complete list of the internal **vi** variables and their meanings is included in the **vi** manual page.

Summary of vi commands

The following tables contain all the basic **vi** commands and variables. Complex commands are omitted; see the **vi** manual page for details.

Table 13-1 Entering vi

Typing this:	does this:
<code>vi file</code>	starts at line one
<code>vi +n file</code>	starts at line <i>n</i>
<code>vi + file</code>	starts at last line
<code>vi +/pattern file</code>	starts at <i>pattern</i>
<code>vi -r file</code>	recovers <i>file</i> after a system crash

Table 13-2 Cursor movement (command mode)

Pressing this key:	does this:
<code>h</code>	moves one space left
<code>l</code>	moves one space right
<code><Space></code>	moves one space right
<code>w</code>	moves one word right
<code>b</code>	moves one word left
<code>k</code>	moves one line up
<code>j</code>	moves one line down
<code><Enter></code>	moves one line down
<code>)</code>	moves to end of sentence
<code>(</code>	moves to beginning of sentence
<code>}</code>	moves to beginning of paragraph
<code>{</code>	moves to end of paragraph
<code><Ctrl>U</code>	scrolls up half a window
<code><Ctrl>D</code>	scrolls down half a window
<code><Ctrl>F</code>	scrolls down one whole window
<code><Ctrl>B</code>	scrolls up one whole window

Table 13-3 Inserting text (command mode, enters insertion mode)

Pressing this key:	starts insertion:
i	before the cursor
I	before first character on the line
a	after the cursor
A	after last character on the line
o	on next line down
O	on the line above
r	on current character, replaces one character only
R	on current character, replaces until (Esc)

Table 13-4 Delete commands (command mode)

Command	Function
dw	deletes a word
d0	deletes to beginning of line
d\$	deletes to end of line
3dw	deletes three words
dd	deletes the current line
5dd	deletes five lines
x	deletes a character
5x	deletes five characters

Table 13-5 Change commands (command mode, enters insertion mode)

Command	Function
cw	changes one word
3cw	changes three words
cc	changes current line
5cc	changes five lines

Table 13-6 Search commands (command mode)

Command	Function	Example
/and	finds the next occurrence of <i>and</i>	and, stand, grand
?and	finds the previous occurrence of <i>and</i>	and, stand, grand
/^The	finds next line that starts with <i>The</i>	The, Then, There
/[bB]ox/	finds the next occurrence of <i>box</i> or <i>Box</i>	
n	repeats the most recent search, in the same direction	

Table 13-7 Search and replace commands (command mode)

Command	Result	Example
:s/pear/peach/g	all <i>pears</i> become <i>peach</i> on the current line	<i>pear</i> becomes <i>peach</i> if present on the current line
:1,\$s/file/directory	replaces first instance of <i>file</i> on each line with <i>directory</i> from line 1 to the end	<i>filename</i> becomes <i>directoryname</i>
:g/one/s//1/g	replaces every occurrence of <i>one</i> with 1	<i>one</i> becomes 1, <i>oneself</i> becomes <i>1self</i> , <i>someone</i> becomes <i>some1</i>

Table 13-8 Pattern matching: special characters (regular expressions)

This character:	Matches:
^	beginning of a line
\$	end of a line
.	any single character
[...]	a set of characters (represented by "..."; the range can either be specified, like [abc] or a range like [a-b])

Table 13-9 Leaving vi (command mode)

Command	Result
<code>:w</code>	writes out the file
<code>:x</code>	writes out the file, quits vi
<code>:wq</code>	writes out the file, quits vi (like <code>:w :q</code>)
<code>:q!</code>	quits vi without saving changes
<code>!<i>command</i></code>	executes UNIX <i>command</i>
<code>:sh</code>	starts a new shell
<code>!!<i>command</i></code>	executes <i>command</i> and places output on current line
<code>:e <i>file</i></code>	edits <i>file</i> (save current file with <code>:w</code> first)

Table 13-10 Options (`:set option`)

This option:	does this:
<code>all</code>	lists all options
<code>term type</code>	sets terminal to type
<code>ignorecase</code>	ignores case in searches (on or off)
<code>list</code>	displays <code><Tab></code> and end-of-line characters (on or off)
<code>number</code>	displays line numbers (on or off)
<code>report</code>	prints number of lines changed by a line-oriented command
<code>terse</code>	shortens error messages (on or off)
<code>warn</code>	turns off "no write" warning before escape (on or off)
<code>magic</code>	allows inclusion of special characters in search patterns without a preceding backslash (on or off)
<code>wrapscan</code>	prevents searches from wrapping around the end or beginning of a file (on or off)
<code>mesg</code>	permits display of messages sent to your terminal with the <code>write</code> command (on or off)

Using ed

`ed` is a line editor; it edits a single line at a time. It makes no use of the terminal, and the cursor movement keys associated with `vi` have no effect.

Instead of showing you a window of text, `ed` relies on *line addresses*. A line address is the number of a line in a file, to which a command is applied. Commands in `ed` may all have zero, one- or two-line addresses. (In the latter case, the two addresses correspond to a range of lines within which the command is carried out.)

Starting ed

From the shell prompt, type `ed filename`, where *filename* is the file to edit. `ed` starts, then displays the number of lines it has read from *filename*. You are in command mode.

Leaving ed

`ed` uses `vi`-like commands (of the type prefixed by a colon (:)). (`vi` is descended from `ed`.) To quit `ed`, use most of the `vi` commands except `ZZ`. Commands are not prefixed with a colon. For more information, see "Stopping `vi`" (page 110) or the `ed` manual page.

Reading a file

To see the contents of your file, use the `l` (list) command. `list` requires line addresses. If no addresses are given, `ed` just displays the current line.

To see lines one to ten of a file, type:

```
1,10l
```

The first digit is the start address, and the second digit is the stop address for the command `l` (list). `ed` displays everything from the start address to the stop address.

To refer to the current line, use "." (a period); this is the address of the current line.

To refer to the last line of the file, type "\$".

You can use *relative* addresses; for example, `$-5` means the fifth line before the last line of the file, while `.+2` means the second line after the current line.

For convenience, a comma (,) stands for the address pair 1,\$ (that is, the entire file), while a semicolon (;) stands for the address pair .,\$ (current line to end of file). For example, to list the entire file, type ,l.

Editing text

There are several commands for editing text in ed:

- c Change text: this command uses an address. The specified lines are deleted, then whatever you type replaces them. When you have finished entering text, press <Ctrl>D to return to command mode.
- i Insert text: this command uses an address. Whatever you type is inserted before the specified line; press <Ctrl>D to stop inserting text.
- d Delete text: this command uses an address. The lines you specify are deleted.
- s/*old*/*new* Replace text: this command searches each addressed line for *old*, and the first occurrence is replaced with *new*. *old* can be a wildcard, as with vi.

If you supply a range of addresses to this command, each line in the range is searched and the first instance of *old* on each line is replaced with *new*.

ed supports most of the same wildcards as vi. For full details, see the ed manual page.

Using sed

sed is a stream editor; it cannot be used interactively like **vi** or **ed**. Commands to **sed** are entered in a script file. **sed** reads a line of input, executes all the applicable commands in its script, and writes the result to its output. It repeats this cycle until there is no more input.

Running sed

To start **sed**, use the following command:

```
sed -f script_name < input > output
```

where *script_name* is the name of a file containing a script of instructions, and *input* and *output* are the input and output files.

sed will execute the script until there is no more input. It will then exit.

If you want **sed** to execute a short command on the contents of a file, you can enter the commands on the command line using the **-e** option:

```
sed -e 'sed commands' input > output
```

sed commands

sed commands are similar to those of **ed**. However, **sed** does not allow relative line addresses, because **sed** never backs up; it reads through the input file just once, from start to finish.

For a full description of the **sed** commands, see the **sed** manual page.

INSTALACIÓN Y MANEJO DE REDES (LAN) EN PLATAFORMA UNIX

5.- ADMINISTRACIÓN DE UNIX EN RED



Mayo de 1996.

Chapter 1

Networking overview

This guide provides background and reference material that helps you configure and maintain the various networking services of Open Desktop.

This chapter describes networking in general and Open Desktop Networking services in particular. After you read this chapter, you will have a better understanding of the components that make up the Networking package, and you can pick and choose which components you want to configure. We strongly recommend that you read this chapter before installing any networking software.

The remaining chapters in this guide describe the networking products that make up the Open Desktop networking package: SCO[®] TCP/IP, SCO NFS[™], SCO LAN Manager Client, UUCP, and MMDF.

- Chapters 2 through 8 describe TCP/IP, including basic maintenance, configuring serial line communications, using the BIND name server, setting up gateways, using the Simple Network Management Protocol (SNMP), setting up remote printing, and using network time protocols.
- Chapters 9, 10, and 11 cover NFS, including basic maintenance, enabling and using the Network Information Service (NIS), and configuring the Automounter.
- Chapter 12 describes LAN Manager Client and describes how to share resources on your network between machines running LAN Manager or SCO XENIX-NET.
- Chapter 13 covers `uucp` and its use in transferring files and performing remote command execution.
- Chapters 14 and 15 describe the two supported mailers provided by Open Desktop: MMDF and `sendmail`.

- Chapter 16 explains how to configure and administer modems and modem lines on your system.
- Chapter 17 provides information on how to solve some typical networking problems.

Within each of these chapters, you are often referred to manual pages for further information. All of the networking manual pages referred to in this guide are available online through use of the `xman(X)` or `man(C)` commands.

Introduction to networking

Networking, simply put, is connecting your computers together so they can share information. Effective networking increases productivity by using computer resources, such as files, printers, and memory, more efficiently. A network puts the power of all of your system's hardware and software at your fingertips.

Although there are many different types of networks, they fall into two general categories: local area networks (LANs) and wide area networks (WAN).

A LAN connects computers that are in the same office or in adjacent buildings. All the computers on a LAN are connected to a single cable. A computer on a LAN can communicate directly to any other computer on that LAN. One LAN may also be connected to another LAN via a gateway computer.

A WAN connects computers that can be as close as several hundred feet to as far as across the globe. These connections are made using phone lines and sometimes satellite connections, if the distance is great enough. Sometimes a computer must go through one or more computers, or gateways, to reach the one with which it wants to communicate.

Most networks are a combination of local and wide area networks. Figure 1.1 displays a portion of a typical local area network. It includes several client computers, a server computer, a printer that is accessible to any machine on the network, an Ethernet™ cable connecting the machines, and a computer running Open Desktop, which may be a client, a server, or both.

Servers, often the most powerful computers on the network, store data that they make available to clients, other machines on the network that have access to the servers' resources. You can have one or more servers on a network, and a machine can be both a client and server. For example, one machine can serve personnel information while another serves sales data. Each machine is, therefore, a server, but each may also be a client to the other machine's data.

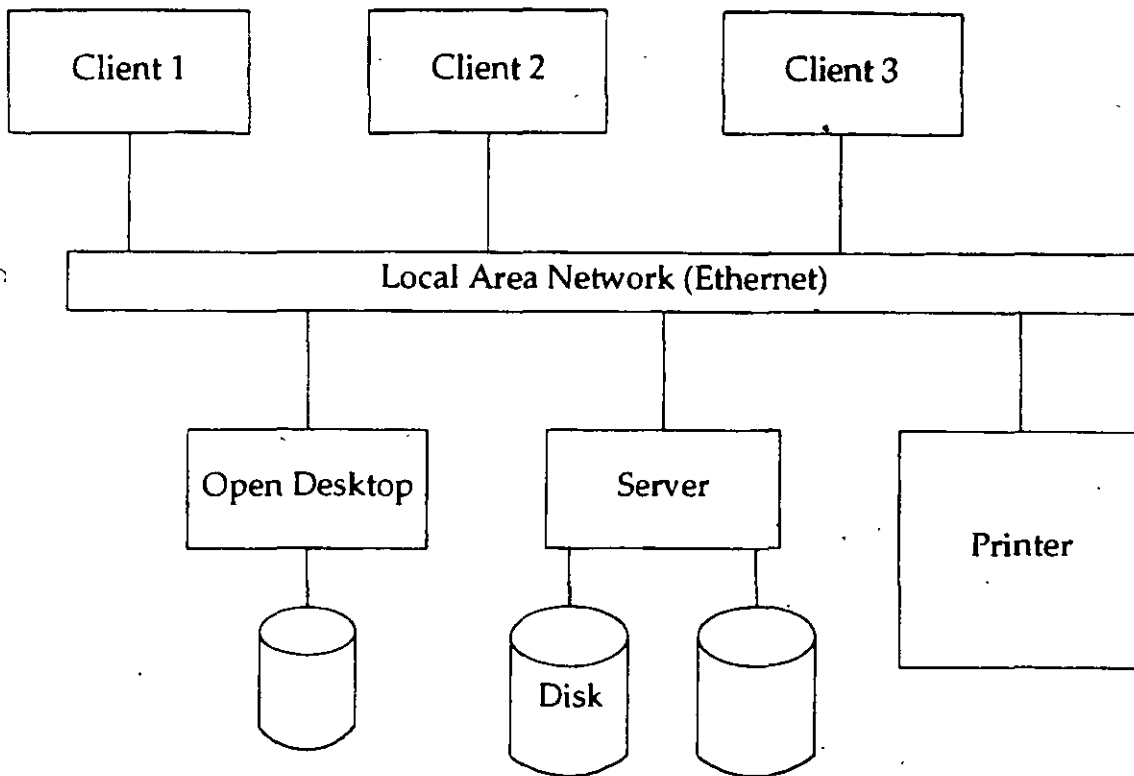


Figure 1-1 Sample Network

How the network works

A network, in the physical sense, consists of cables or phone lines. These lines connect the computers, and networking cards provide the means to talk across them. However, a network is not useful unless it has programs on each computer that let humans access the various computers on the network.

Computers on a network have agreed ways of communicating called protocols. Protocols dictate which signals computers use across cables, how they tell one another that they have received information, and how they exchange information.

Protocols are more accurately termed protocol suites or protocol families. This subtle shift in terminology reflects the fact that the communications functions are complex and are usually divided into independent layers, also called levels. The protocol associated with each layer communicates with only the layers immediately above and below it, and assumes the support of underlying layers.

In protocol suites, lower layers are closer to the hardware and higher layers are closer to the user. The number of layers and tasks that the layers perform depends on who defines them. The following sections describe several popular definitions of protocols and how they relate to each other.

OSI

The International Organization for Standards (ISO) has a seven-layer protocol called the Reference Model of Open Systems Interconnection (OSI), as shown in Table 1-1. OSI also serves as a basis for the X.25 protocol.

Table 1-1 ISO Model

Layer	Name
7	Application
6	Presentation
5	Session
4	Transport
3	Network
2	Data Link
1	Physical

The layers operate as follows:

Layer	Task
Application	defines how programs can communicate with each other
Presentation	performs any necessary data conversion
Session	establishes an enhanced connection (session) with other machines
Transport	makes sure information exchanged between computers arrives intact and without errors
Network	makes sure information coming from one computer arrives at the correct destination
Data Link	makes sure information gets from one end of the cable to the other intact
Physical	defines the way information travels on a cable. A common physical-level protocol is the IEEE 802.3 protocol, of which Ethernet is a subset.

TCP/IP

Transmission Control Protocol with Internet Protocol (TCP/IP) has four software layers built on an underlying hardware layer. Its model is shown in Table 1-2.

Table 1-2 TCP/IP Model

Layer	Name
4	Application
3	Transport
2	Internet
1	Network Interface

The layers operate as follows:

Layer	Task
Application	accesses the Internet, and sends and receives data
Transport	provides communication between application programs
Internet	takes care of communication among machines
Network Interface	accepts and transmits data over the network

Correspondence of models

Table 1-3 shows how the ISO and TCP/IP models correspond to each other.

Table 1-3 Protocol Families

ISO		TCP/IP	
Layer	Functionality	Functionality	Layer
7	Application	Application	4
6	Presentation		
5	Session		
4	Transport	Transport	3
3	Network	Internet	2
2	Data Link	Network Interface	1
1	Physical	Hardware	—

Designing or adding to your network

Your Open Desktop machine may be a part of an entirely new network, or it may become a machine on a network that already exists. In either case, you need to make several decisions about your Open Desktop machine:

- With what other computers does it need to communicate?
- Will it serve as a client, a server, or both?
- Who will use this machine, and what sort of access do they need?

Common networking administration tasks

After you decide how your Open Desktop machine fits into the network, you need to install and configure the appropriate Networking services as described in the Open Desktop *Installation and Update Guide*. You also need to update the networking files on other machines so that they know of the new

machine's existence. This configuration ensures, among other things, that:

- all machines on the network know each other's names and addresses
- individual users will have access to files and accounts on various machines
- electronic mail is routed correctly
- the network runs at peak efficiency

Common tasks that you will perform to ensure these goals include:

- installing and maintaining networking hardware and software
- assigning names and addresses to each computer and device on the network
- assigning names and identification numbers (IDs) to network users and groups
- performing the commands required to share, remove, and restrict resources
- updating all appropriate networking files on your network's machines

The remainder of this chapter describes the various Networking services used by Open Desktop, and provides pointers to the rest of this guide.

Open Desktop networking services

The Open Desktop networking package consists of several services that allow your computer to connect to both local and wide area networks.

- TCP/IP provides various protocols for networking communications that other networking packages, such as NFS, use. The TCP/IP package also provides end-user programs such as *telnet* and *ftp* that enable remote logins and file transfer between systems running TCP/IP.
- NFS allows you to export filesystems to the network so that users on other computers can use them as if they were local, and to import filesystems from remote servers to your Open Desktop client.
- LAN Manager Client provides a way to access resources (files, programs, printers, and so on) remotely from other machines running compatible versions of LAN Manager and XENIX-NET.
- UUCP enables you to execute commands remotely and to transfer files and mail between remote UNIX systems connected by serial lines.
- MMDF controls electronic mail communications between machines running the UNIX Operating system.

The remainder of this chapter introduces each of these services. It briefly explains the issues involved in configuring and maintaining each service and describes many of the files and utilities that each service uses.

Introducing TCP/IP

TCP/IP is the set of protocols and programs used by Open Desktop to interconnect computer networks and to route traffic among different types of computers. "TCP" stands for Transmission Control Protocol, and "IP" stands for Internet Protocol. These protocols describe allowable formats, error handling, message passing, and communication standards. Computer systems that use TCP/IP are thus able to speak a common language, despite any differences in the hardware and software of the various machines.

Many large networks conform to these protocols, including the DARPA Internet (Defense Advanced Research Projects Agency Internet). A variety of universities, government agencies, and computer firms are connected to an internetwork that follows the TCP/IP protocols. Thousands of machines are connected to this internet, or network of networks. Any machine on the internet can communicate with any other. Machines on the internet are referred to as *hosts* or *nodes*, and are defined by their internet (or IP) address. Defining an internet address is described later in this section.

TCP/IP provides the basis for many useful services, including electronic mail, file transfer, and remote login. Electronic mail is designed to transfer short text files. The file transfer application programs transfer very large files containing programs or data. They also provide security checks controlling file transfer. Remote login allows users on one computer to log in at a remote machine and carry on an interactive session. The TCP/IP programs that facilitate these services are described in detail in the Open Desktop *User's Guide*.

The internet protocol (IP)

The Internet Protocol, IP, defines a data delivery system wherein the sending and receiving machines are not necessarily directly connected. IP splits data into packets of a given size, which are then forwarded to the receiving machine via the network. These individual packets of data (often called *datagrams*) are routed through different machines on the internet to the destination network and receiving machine. A particular set of data, such as a file, may be broken up into several datagrams that are sent separately. When you use IP to forward datagrams, individual datagrams may or may not arrive, and they probably will not arrive in the order in which they were sent. TCP adds the reliability that IP lacks.

A datagram consists of header information and a data segment. The header information is used to route and process the datagram. Datagrams may be further fragmented into smaller pieces, depending on the physical requirements of the networks they cross. For example, when a gateway sends a datagram to a network that cannot accommodate the datagram as a single packet, the datagram must be split into pieces that are small enough for

transmission. The datagram fragment headers contain the information necessary to reassemble the fragments into the complete datagram. Fragments do not necessarily arrive in order; the software module implementing the IP protocol on the destination machine must reassemble the fragments into the original datagram. If any fragments are lost, the entire datagram is discarded.

The transmission control protocol (TCP)

The Transmission Control Protocol, TCP, works with IP to provide reliable delivery. It provides a means to ensure that the various datagrams making up a message are reassembled in the correct order at their final destination and that any missing datagrams are resent until they are correctly received.

The primary purpose of TCP is to avoid the loss, damage, duplication, delay, or misordering of packets that can occur under IP. Also, security provisions such as limiting user access to certain machines can be implemented through TCP.

TCP provides reliability using checksums (error detection codes) on the data, sequence numbers in the TCP header, positive acknowledgment of data received, and retransmission of unacknowledged data.

Other TCP/IP protocols

The protocols listed in Table 1-4 are provided as part of TCP/IP:

Table 1-4 Additional TCP/IP protocols

Protocol	Purpose
Address Resolution Protocol (ARP)	ARP translates between DARPA Internet and Ethernet addresses.
Internet Control Message Protocol (ICMP)	ICMP is an error-message and control protocol used by TCP/IP.
Point-to-Point Protocol (PPP)	PPP provides both synchronous and asynchronous network connections.
Reverse Address Resolution Protocol (RARP)	RARP translates between Ethernet and DARPA Internet addresses.
Serial Line Internet Protocol (SLIP)	SLIP enables IP over serial lines.
Simple Mail Transport Protocol (SMTP)	SMTP is used by MMDf to send mail via TCP/IP.
Simple Network Management Protocol (SNMP)	SNMP is the protocol used to perform distributed network management functions via TCP/IP.
User Datagram Protocol (UDP)	UDP provides data transfer without many of the reliable delivery capabilities of TCP. UDP is less CPU-intensive than TCP, and is useful when guaranteed data delivery is not of paramount importance.

These protocols are described in further detail later in this guide.

TCP/IP end-user commands

Several TCP/IP commands, described in detail in the *Open Desktop User's Guide*, provide end users with networking capabilities. Table 1-5 is a partial list of these commands:

Table 1-5 TCP/IP Commands

Command	Purpose
ftp	file transfer between machines running TCP/IP; these machines may or may not run the same operating system
rcmd	remote command execution on another UNIX machine
rcp	file copying between two UNIX machines
rlogin	remote login on another UNIX machine
ruptime	status display of local network machines
rwho	displays list of users logged on to local network machines.
telnet	remote login on a machine running TCP/IP; these machines may or may not run the same operating system

Configuring TCP/IP

This section provides information on software and hardware prompts you need to answer as you configure TCP/IP. We strongly recommend that you read and understand this section before you fill in the configuration checklists located in the *Installation and Update Guide*. Installation prompts include:

- system's host name and domain name
- Internet address(es) for each driver, adapter, or serial line
- broadcast address
- netmask
- hardware interrupt vectors, base memory addresses, and RAM buffer sizes

System name

Your system name, or host name, should be unique on your network. It can consist of lowercase letters and numbers, must begin with a letter, and should be no longer than eight characters. mail and other programs use the system name to identify the correct data destination. Here are some sample valid machine names: *scosysv*, *tcpdev*, *account1*.

Driver type

The driver is the software that allows your networking cards or hardware to interact with TCP/IP. Each card, adapter, slip line or ppp line that you use must be uniquely associated with a particular device driver. You can install up to four Ethernet cards of one type, up to two Token Ring adapters, and up to eight serial line interfaces (four SLIP and four PPP), but you can only configure one driver at a time. When you are prompted for the driver type, choose the type you want to configure.

Interrupt vector

Each driver on your system, including those for network cards and SLIP lines, must have its own interrupt vector, or IRQ. This vector must not be used by any other device on the system. Refer to your networking hardware documentation to determine what vectors the hardware supports. In addition, the `hwconfig(ADM)` and `vectorsinuse(ADM)` programs list the hardware already installed on your system and what vectors are already in use, respectively.

Your networking hardware might be pre-configured to use a particular vector. If you want to change this vector setting, you might also need to change the physical jumper settings on the board or run a setup program provided with the board.

NOTE A number of networking cards are pre-configured to use interrupt vector 3. Your operating system has reserved IRQ3 for the sio (serial input-output) device. You can either disable this device during your `netconfig` session, or choose another vector.

I/O base address

Each hardware driver on your system that performs I/O (input/output) needs a unique memory base address so that the system can locate it. This memory address is a three- or four-digit hexadecimal number, must match the settings on the board, and must not conflict with any other hardware on your system. Valid base addresses are displayed when you configure your card.

Thick/thin cable

Some networking cards use thick, rather than thin, networking cable.

- Thin cable provides a direct connection to the network without the use of a transceiver. Most installations use thin cable.
- Thick cable connects your networking card to a transceiver, which in turn connects to the Ethernet cable.

RAM buffer size and base address

Several networking cards require a designated space in RAM to do buffering; you need to specify this address (as a five-digit hexadecimal number) and, if necessary, configure the buffer size.

NOTE The `wdnsetup` command is used to change these values for some Western Digital cards. For more information on this command, see the `wdnsetup(ADM)` manual page.

Domain name

The MMDF mail router uses the domain name to route messages, such as mail, from machine to machine. The domain name allows your network to fit into a hierarchical network structure composed of commercial organizations (.COM), educational institutions (.EDU), the government (.GOV), the military (.MIL) or miscellaneous organizations (.ORG). Sample domain names are *sco.COM* (the domain name used by SCO) and *berkeley.EDU* (the domain name used by the University of California at Berkeley).

Base your domain name choice on the following:

- If other machines on your network already use a domain name, use the same name for the machine you are installing.
- If you are creating a new domain and want to use BIND to connect to the outside world, you need to register the name with the appropriate network (DARPA Internet, CSNET, or BITNET). To register a domain name, write to:

DDN Network Information Center
Suite 200
14200 Park Meadow Drive
Chantilly, VA 22021
- If you are creating a new domain and are not currently connecting to an outside network, use the name `.UUCP`.
- If you will never attach to a network outside your company, choose *company.COM*.

IP address

The IP address identifies and differentiates your machine from all others on the network. It consists of a 32-bit binary number that is usually displayed as four octets expressed in decimal and separated by periods. You must have a unique IP address for each machine on your network. In addition, if your machine serves as a *router* to another network, it contains two or more network cards and belongs to two or more networks. In this case, you must assign each card a unique IP address on the appropriate network.

NOTE The IP address differs from an Ethernet address in that it is configurable. An Ethernet address is a 6-byte address that is unique to each physical Ethernet card. This non-configurable address is assigned by the card manufacturer.

The IP address consists of two parts: a network address that identifies the network and a host address that identifies the particular host, or node. Table 1-6 shows an IP address in binary form, as binary octets, as decimal octets, and as it appears in standard notation.

Table 1-6 IP address derivation

binary (32-bit)	10000100100011110000001000000010			
binary (octets)	10000100	10001111	00000010	00000010
decimal octets	132	147	2	2

IP address (in standard notation) = 132.147.2.2

Several classes of TCP/IP networks are available, each based on the number of hosts a network needs. Network classes supported by SCO are Class A, B, and C. Use the smallest network class that can accommodate all of your network's hosts. Most TCP/IP installations use Class C, but some larger installations might need to use Class B.

Table 1-7 lists valid network addresses for each class:

Table 1-7 Internet address classes

Class	Available Hosts per Network	Valid Address Ranges
A	16777216	1.0.0.1 through 126.255.255.254
B	65534	128.0.0.1 through 191.255.255.254
C	254	192.0.0.1 through 222.255.255.254
Reserved		224.0.0.0 through 255.255.255.254

If you are connecting your machine to a pre-existing network, the network address (for Class A, the first octet; for Class B, the first two octets; and for Class C; the first three octets) is the same as those of other machines on the network. In this case, you need only concern yourself with creating a unique host address.

If you are creating an entirely new network and you want to connect to the DARPA Internet, you need to contact the Network Information Center to have a network address assigned. The full address is shown earlier in the section "Domain name". If you do not want to connect to an outside network, you

can choose any network address as long as it conforms to the syntax shown previously. In either case, once you determine the network address, you can then create the unique host address.

When you determine the IP address, keep in mind the following:

- Each logical network must have its own network address.
- All hosts in a network must have the same network address.
- All hosts in a network must have unique host addresses.
- Do not use the following network addresses: 0 or 127 (Class A), 191.255 (Class B), 223.255.255 (Class C), or any of the addresses shown in the Reserved class of Table 1-5.

Broadcast address parameters

All datagrams sent by TCP/IP move through all machines in the network path. However, each host adapter ignores any packet that does not include that particular computer's IP address in the datagram header. Occasionally, you might want to send a message to all machines on a particular network. To do so, select a *broadcast address* for your machine. A broadcast address is one in which the host portion of the IP address consists either of all 0's or all 255's. The configuration procedure prompts you to choose between the following address schemes:

Table 1-8 Broadcast address schemes

Scheme	Example	Purpose
all zeroes (decimal 0)	132.147.0.0	provides compatibility with 4.2BSD systems
all ones (decimal 255)	132.147.255.255	UNIX Operating System Standard. (RFC-919)

The addresses shown in the previous table are for a class B network, and are shown as examples only. Your values will be different. If you are on a network that does not contain any machines running 4.2BSD UNIX or earlier BSD versions, choose all ones. If such machines exist on your network, choose all zeroes.

netmask setting

The netmask strips the network ID from the IP address, leaving only the host ID. Each netmask consists of binary ones (decimal 255) to mask the network ID and binary zeroes (decimal 0) to retain octets of the host ID of the IP address. For example, the default netmask setting for a Class B address is 255.255.0.0.

NOTE Always use the default netmask that the installation program prompts you for unless you are creating a subnet, a logical division of a physical network. If you create a subnet, also mask the portion of the address that indicates the subnet. For example, a netmask for a subnet of a Class B address is 255.255.255.0. For more information on creating subnets, see the chapter "TCP/IP network administration" later in this guide.

TCP/IP over a SLIP or PPP serial line

The following prompts are relevant only to serial line drivers.

tty line

This line indicates what *tty* the SLIP line connects to.

- If you are connecting to COM1:, interrupt vector 4, enter *tty1A*.
- If you are connecting to COM2:, interrupt vector 3, enter *tty2A*.
- If you are connecting to a smart serial card, use the appropriate *tty* naming convention.

Source IP address

Enter the IP address for this host (this end of the serial line). For more information on determining IP addresses, see "IP address" earlier in this chapter.

Destination IP address

Enter the IP address for the remote host (the opposite end of the line).

Baud rate

Enter the baud rate at which data is transmitted. The default is 9600.

SLIP netmask

Choose a netmask for this SLIP line. For more information on netmasks, see the section "netmask setting" earlier in this chapter.

Maintaining TCP/IP

After you install TCP/IP, you may never need to alter the TCP/IP configuration again. However, there are some common tasks that occur if you want to customize or add to your network. These are described briefly here and in detail later in this guide.

Adding hosts

The */etc/hosts* file is a list of hosts on the network. Network library routines and server programs use this file to translate between host names and Internet addresses when the BIND (Berkeley Internet Name Domain) name server is not being used.

To add a machine to the network, you must add an entry to all of the */etc/hosts* files on the local network. Refer to the *hosts(SFF)* manual page for a description of the file format.

Configuring the name domain server

The Berkeley Internet Name Domain Server (BIND) provides a distributed lookup system for host names and addresses. Enabling BIND overrides the default network information file, */etc/hosts*. For more information, see the chapter "Configuring the BIND name server" later in this guide.

Setting up routing tables

Routing tables provide the information needed to route packets to their destinations properly. For descriptions of several possible approaches to maintaining routing information, see the chapter "Gateways and routing" later in this guide. In addition, the chapter "TCP/IP network administration" contains a section on obtaining information about the system routing tables.

Establishing user equivalence

You can control who has access to a machine through the network by establishing user equivalence within the */etc/hosts.equiv* and *.rhosts* files. The *rlogin*, *rcp*, and *rcmd* commands use these files to verify access privileges. For information on how to use these files, see the section "Network databases" in the chapter "TCP/IP network administration" later in this guide. You can also refer to the *hosts.equiv(SFF)* manual page for a description of the file format. A note in the "Helpful Hints" chapter also discusses user equivalence.

Setting up anonymous ftp

You can set up a public *ftp* account on your system that allows remote users to transfer files anonymously from restricted, public directories on your system. For information on the */etc/ftpusers* file, and a description of how to set up the public *ftp* account, refer to the section "Network databases" in the chapter "TCP/IP network administration" later in this guide.

Altering installation parameters

You can change many of the settings that you set during TCP/IP installation by altering the appropriate system files (such as */etc/hosts* and device driver files) with a text editor or with an appropriate utility, such as **netconfig**, **route**, or **mkdev**. The use of such files and utilities, which is documented in the chapter "TCP/IP network administration," is always preferable to reinstalling the software.

Tuning kernel parameters

You may need to tune kernel parameters by increasing or decreasing STREAMS buffers and other parameters used by TCP/IP. Several utilities, including **netstat**, **configure**, and **netconfig**, help you fine-tune your system to enhance networking performance. These utilities are described in the chapter "TCP/IP network administration."

Monitoring TCP/IP status

You can use the **netstat** command to display Internet connections, current Internet activity, routing tables, and error messages, among other useful information. In addition, you can use the Simple Network Management Protocol (SNMP) commands and utilities to further monitor and troubleshoot your network. For more information, see the chapter "Configuring and using SNMP" later in this guide.

Enabling remote printing

You can enable the remote printing daemon, **lpd**, to allow print jobs to be sent over the network to remote printers, or to make a printer attached to your computer available to the network. For information on **lpd** and its associated files, see the chapter "Remote line printing" later in this guide.

Introducing NFS

SCO Network File System (NFS) is a product that enables you to export and mount filesystems across a network. This allows users on a local machine, or client, to access specified files and directories from a remote machine, or server, without the time-consuming process of remote logins or machine-to-machine file copying.

NFS also contains the Network Information Service (NIS), which enables you to perform common account administration tasks from one NIS server.

Because NFS fits into the application layer of TCP/IP, it can communicate between many different types of hardware and operating systems, as long as these systems run TCP/IP.

NFS makes use of the following files, commands, and daemons, each of which are described in detail in Chapter 9, "Administering NFS," of this guide.

- The file */etc/exports* lists the exportable filesystems on a given server.
- The file */etc/hosts* informs TCP/IP, and hence NFS, how to find a given server or client.
- The file */etc/default/filesys* lists the remote filesystems a client may mount and on what server the client may find them.
- The **mount** command mounts an exported filesystem on the client. This command invokes the **mountd** and **nfsd** daemons.

Maintaining NFS

If you want to export filesystems with NFS, you must create */etc/exports* after installing Open Desktop. A sample file is shown in Chapter 9, "Administering NFS." In addition to creating */etc/exports*, you also must ensure that the following conditions are true:

- The */etc/hosts* file on each server and client must contain the names and IP addresses of each server and client you want to access.
- The */etc/default/filesys* file on each client must contain the names and locations of all of the filesystems that you want to access.

After you edit these files to satisfy these conditions, you may want to fine-tune NFS as shown in Chapter 9.

Introducing LAN Manager client

LAN Manager Client is a program running under Open Desktop that allows other computers on a LAN that run compatible servers to share resources. Resources can be any of the following:

- files
- directories
- programs
- printers
- plotters
- tape drives

LAN Manager is designed so that users can access files and printers on other computers running LAN Manager simply by typing a command on their own computer. Because LAN Manager runs on the DOS and UNIX systems as well as Open Desktop, you can have PCs, multiuser UNIX systems, and Open Desktop workstations sharing resources on the same LAN.

To use LAN Manager correctly between Open Desktop and UNIX systems, you must maintain user (UID) and group (GID) consistency between the machines on your network. If you are connecting existing computers in a network for the first time, be certain that user IDs are consistent from computer to computer. If you are adding your Open Desktop machine to an existing network, you should create user accounts and groups with IDs that match those found throughout the network. User IDs are defined by the `sysadmsh(ADM) Accounts ⇨ User ⇨ Create` selection and placed in UNIX system database files, including `/etc/passwd`. For information on adding user accounts, see the "Administering User Accounts" chapter in the *Administering Operating System Services* portion of this guide. For information on maintaining ID consistency, see the `passwd(C)`, and `passwd(F)` manual pages. When you create new user accounts or change account information, keep in mind the following:

- Always use the `sysadmsh(ADM) Accounts ⇨ User ⇨ Create` selection to administer accounts. Never edit `/etc/passwd` with a text editor. This could damage the password database information and cause your system to refuse further logins.
- User ID numbers must not be duplicated among different users. An individual user can have several different accounts on different machines, but to maintain user equivalency each account must have the same user name and ID number.

Configuring LAN Manager Client

When you install LAN Manager Client, the custom utility configures LAN Manager Client and adds your computer to the network. Before LAN Manager Client is ready to use, you must configure it with `netconfig(ADM)` as described in the *Installation and Update Guide*.

When you enable LAN Manager Client with `netconfig`, you are prompted to confirm the hostname and confirm that LAN Manager Client starts up each time the system is brought into multiuser mode. The most important prompt, however, asks you to provide a NetBIOS scope identifier for your LAN Manager network.

All machines on your network must use the same scope identifier to successfully communicate with one another. Choose one based on the following guidelines:

- If you are adding this machine to an existing network, use the same scope identifier that is used on any other machine on the network. On a SCO UNIX or Open Desktop machine, the scope identifier is found in the file `/etc/default/nbconf` as the variable `NB_SCOPE`.
- If you are creating a new network that contains only SCO UNIX or Open Desktop machines, use the domain name (such as `sco.COM`) minus its suffix. For example, if your domain name is `mycomp.COM`, enter `mycomp`.
- If you are creating a new network that also contains DOS or OS/2 machines, it is a good idea to set the scope identifier to `NULL`.

NOTE To enter a `NULL` scope identifier during `netconfig` configuration, enter two double quotes ("") at the prompt.

You can enter a scope identifier or choose the default during your `netconfig` session, or you can change the scope identifier later by editing the file `/etc/default/nbconf`. To change the scope identifier in this file, change the `NB_SCOPE` variable to the desired scope. To set this variable to `NULL`, edit the variable line so that it reads:

```
NB_SCOPE=""
```

If you use this procedure to change `NB_SCOPE`, you must reboot the machine before this change takes effect.

Maintaining LAN Manager Client

A common activity that you may perform as a system administrator is configuring LAN Manager Client so that it runs more efficiently. You do so in two ways: by altering system-wide configuration parameters and relinking the kernel, or by changing default values found in the networking startup file,

/usr/lib/xnet/xnetrc. For example, you might decrease the number of network buffers available, making more memory available for user programs. For information on these maintenance issues, refer to Chapter 12, "LAN Manager client administration."

Introducing UUCP

UUCP, the UNIX-to-UNIX copy program, is used when your Open Desktop machine's network connection consists of a serial line and modem connection to another machine. If your network consists of an Ethernet LAN, with or without an Internet connection, you do not need to use UUCP, as it duplicates many features found in LAN Manager Client and TCP/IP.

The following UUCP features are available:

- file transfer between your machine and a remote UNIX machine, or between remote UNIX machines, with the **uucp** command; this command is comparable to the **ftp** and **rcp** commands
- remote command execution with the **uux** command; this command is comparable to the **rcmd** command
- mail delivery to and from remote sites with the **mail** command via the UUCP modem and phone line connection

UUCP spools requests for communication until contact is made with a remote system. You can configure your Open Desktop machine to communicate with other machines rarely, daily, hourly, or even on demand.

UUCP consists of a variety of administrative and end-user programs that help you maintain and use UUCP. These programs are described in detail in Chapter 13, "Building a Remote Network with UUCP," in this guide.

Configuring UUCP

To set up UUCP for use, do the following:

1. Attach a modem and phone line to a serial port on your computer.
2. Use **uuinstall** to configure UUCP on your system by choosing a site name, updating host lists, and configuring dial-in/dial-out lines.

After you perform these actions, you can use UUCP. Several **uudemon** shell scripts and the **uulog** program help you maintain UUCP after configuration by polling sites automatically, displaying status information, cleaning UUCP directories, and so forth. For more information on these programs, refer to Chapter 13.

Introducing MMDF

Open Desktop uses MMDF, the Multi-channel Memorandum Distribution Facility, to route mail locally and to and through channels (such as UUCP) that offer MMDF support. Here are the key facets of MMDF:

- Domain names are used to route messages from machine to machine.
- Channels, such as UUCP, SMTP, or Micnet, are the programs that MMDF uses to move mail from one location to another. Aliases allow you to provide your users with shorthand ways of specifying addresses. For example, the address *info-p1-outbound@berkeley.EDU* could be aliased to *info*.
- A configuration file, */usr/mmdf/mmdftailor*, pulls all of the MMDF pieces together by specifying the local machine's name, domain, company, and UUCP name, as well as describing how, when, and where to contact other machines.

Configuring MMDF

After the appropriate files are installed on your system, you need to run `mkdev mmdf` to configure MMDF. Refer to Chapter 14, "Setting up electronic mail," for more information.

Other networking software

Open Desktop provides two other types of networking software that do not fall directly under any of the packages described earlier in this chapter. They are:

- `sendmail` is another mail routing and distribution program provided with Open Desktop. We recommend that you use MMDF, as it provides an enhanced feature set and is easier to configure than `sendmail`. If you do decide to use `sendmail`, complete instructions on installation and configuration are found in Chapter 15, "Sendmail administration," of this guide.
- `cu(C)` allows you to dial in and dial out of your Open Desktop computer by means of a standard telephone line and modem. In doing so, you can emulate a terminal session on a remote machine, or dial into your Open Desktop machine from a remote terminal or computer. Chapter 16, "Using modems," explains how to install and configure the appropriate hardware on your system to perform these actions.

Chapter 3

Tools reference

The operating system includes tools that measure performance, reconfigure the system for better performance, and analyze other system problems that arise. The system administrator can use these tools to locate problem areas. The tools described in this section are:

- sar** samples the state of the system and provides reports on various system-wide activities.
- swap** with the **-l** option, reports on the amount of available swap space. Use the **-a** option to configure additional swap devices.
- crash** displays various kernel structures. The discussion here concentrates on using **crash** to view how STREAMS resources are being used, to see the current value of many tunable parameters, and to study the state of the system at the time of a system PANIC. **crash** has many other uses that are not discussed here.
- ps** reports on processes currently executing.
- timex** reports on system resource usage during the execution of a command or program.
- configure** a menu-driven program used to modify the value of tunable parameters that are included in the Operating System Services component of Open Desktop
- fsck** checks the integrity of a filesystem and attempts to fix any corruption it finds
- id tune** used to modify some tunable parameters that cannot be modified with **configure**
- integrity** checks the Security system for corruption
- tunesh** an automatic configuration tool that adjusts kernel parameters based on information you supply about your configuration

configure(ADM) — reallocate kernel resources

The `configure` utility is a menu-driven program that presents each tunable kernel parameter and prompts for modification. After modifying kernel parameters, you must relink the kernel by invoking `link_unix(ADM)`, reboot, and test the new kernel. To change any kernel parameter, do the following:

1. Reboot and enter single-user (maintenance) mode.
2. Enter the following commands to start the `configure` program:

```
cd /etc/conf/cf.d
./configure
```

Δ `sysadmsh` users select: System ↵ Configure ↵ Kernel ↵ Parameters

3. The `configure` menu is displayed. The parameters are grouped by category. The meanings of the parameters are discussed in Tunable system parameter descriptions (page 873).

Choose a category by entering the number preceding it. The resources in that category are displayed, one by one, each with its current value. Enter a new value for the resource, or to retain the current value, simply press `<Return>`. After all the resources in the category are displayed, `configure` returns to the category menu prompt. Return to the Main Menu to choose another category or exit `configure` by entering `q`.

4. After you finish changing parameters, you must link them into a new kernel. Enter the following command:

```
./link_unix
```

Δ `sysadmsh` users select: System ↵ Configure ↵ Kernel ↵ Rebuild

This assembles each of the kernel modules into a new kernel, which must now be installed. Linking can take a few minutes.

5. Boot the new kernel with the following command:

```
/etc/shutdown
```

Δ `sysadmsh` users select: System ↵ Terminate

A boot prompt appears. When you press `<Return>` to reboot the system, the new kernel is loaded and run.

If problems exist with the new kernel, reboot `/unix.old` by entering `unix.old` at the `Boot :` prompt.

Using the configure command line

`configure` also has a command-line interface suitable for use by application developers. For instance, a database developer who finds that 70 files rather than 50 files need to be locked simultaneously may provide a shell script to perform the reconfiguration. To find the current value of any configurable resource using the command-line interface, enter:

```
./configure -y RESOURCE
```

Here *RESOURCE* is the name of the tunable parameter (in uppercase). To change the value of any resource from the command line, enter:

```
./configure RESOURCE=value
```

Overriding configure limit warnings

The `configure` utility notifies you if you exceed certain limits on values for kernel parameters. This prevents you from accidentally exceeding a value, which could severely degrade performance in some cases. The `configure` menu program verifies that you do mean to override the limits. You can also use the `-o` option as described on the `configure(ADM)` manual page to override a configuration limit. The override option only works if you are specifying a parameter on the command line; you cannot use it with the menu.

crash(ADM) — view memory

The `crash(ADM)` utility allows you to analyze the core image of the operating system. It is frequently used in postmortem analysis of a system PANIC, but can also be run on an active system. `crash` is not primarily an administrative tool; it is most useful for debugging the kernel after installing drivers and other kernel-level code. Nonetheless, `crash` provides some information that can help you monitor the performance of your system and troubleshoot system problems that arise. This documentation provides information about `crash` reports related to these topics; see the `crash(ADM)` manual page for full information about using `crash`.

Initializing crash

You must be logged in as *root* to run `crash`. The full command to begin the `crash` session is:

```
/etc/crash [-d dumpfile -n namelist] [-w outputfile]
```

The *dumpfile* is `/dev/mem` for an active system (the default), or the name of the file that contains the memory dump you saved when rebooting the system after a system PANIC. *namelist* is the name of the bootable kernel image, usually `/unix` unless you booted the system under a different name.

When **crash** is initialized, you see the following display:

```
dumpfile= /dev/mem, namelist= /unix, outfile= stdout
```

The ">" is the **crash** prompt; enter the **crash** function you want followed by any options you need and <Return>. Note the following:

- The **-w** option redirects the output of the command to the file specified. For example, **-w var_stats** writes the output of **crash var** to the *var_stats* file. You can use **-w** on the **crash** command line to redirect all **crash** output to a file, or you can use **-w** on the function line to redirect the output of that particular function to a file.
- Any **crash** function can be piped to any shell function. For instance, the command **var ! cat >> file** takes the output of the **crash var** function and appends it to the end of *file*; **var ! lp** redirects the output of this same command to the line printer. Note that the exclamation point "!" serves as the pipe symbol from within **crash**.
- Use the **q** function to end the **crash** session.

The **crash stat** command is a good starting place when analyzing a postmortem.dump. **stat** lists off system statistics (such as the system name and time of PANIC) that you can use to verify that you are looking at the right dump. The **crash panic** command will usually contain the last PANIC message that was displayed on the console.

crash strstat (statistics about STREAMS buffers)

Use **crash strstat** to view current usage statistics on the STREAMS buffers for your system. This information can be used to select correct values for the **NBLK n** tunable parameters and other tunable parameters discussed in "STREAMS data" (page 886). The following example shows a sample session using the **crash strstat** utility. The listings are associated with the following parameters:

```
streams          NSTREAM
queues           NQUEUE
data block size  $n$   NBLK $n$ 
```

The listing "message blocks" is a composite number based on the total block allocation; it does not correspond to a kernel parameter. The **NBLK n** data blocks are also known as STREAMS buffers.

Sample crash strstat output

```
* /etc/crash
dumptime = /dev/mem, namelist = /unix, outfile = stdout
- strstat
- ITEM          CONFIG  ALLOC  FREE    TOTAL    MAX    FAIL
streams         168     84     84     1190     86     0
queues         624    380    244    2621    388     0
message blocks 1455    155   1300   1332119  323     0
data block totals 1164    155   1009   1079923  322    198
data block size 4       256     0    256    28571    19     0
data block size 16      168     2    166    10540    52     0
data block size 64      296    29    267    831712   90     0
data block size 128     168    87     81    152605   139    185
data block size 256    104    29     75    18593    83     13
data block size 512     72     8     64    31949    10     0
data block size 1024    48     0     48    2286     8     0
data block size 2048    40     0     40    3601     6     0
data block size 4096    12     0     12     66     1     0

Count of scheduled queues: 0
quit
```

The following table defines the column headings used by the **strstat** display.

STREAMS display headings

Heading	Description
CONFIG	number of items currently configured
ALLOC	number of items currently allocated
FREE	number of items available for allocation. This number is the difference between CONFIG and ALLOC.
TOTAL	number of attempted allocations since system startup
MAX	highest number of items allocated at one time since system startup.
FAIL	number of failures due to insufficient free items since system startup

A non-zero value in the "FAIL" column indicates that a request was made for a buffer and that no buffers of that size were available. STREAMS failures degrade system performance and, in extreme cases, can cause your system to lock up. In this situation, use the **configure(ADM)** command to increase the number of buffers.

var (values of tunable parameters)

The `crash var` command lists the values of the tunable parameters that are listed in the `var` structure. These same values are listed when you run `configure(ADM)` or `sysdef(ADM)`, but the `crash` listing is useful when you want to check a value when you are running `crash`.

fsck(ADM) — check and repair filesystems

The `fsck(ADM)` program should be run on all filesystems after any system crash or abnormal system termination. `fsck` checks filesystem integrity. Because filesystems are not unmounted gracefully when the system crashes, the structure of the filesystem itself as well as user, system, and application files that were in the process of being updated can be damaged. In most cases, `fsck` fixes any filesystem problems; in some cases, `fsck` may be unable to fix a file, but can recover some portion of the file in the *lost+found* directory of the filesystem. The filesystem being checked must be unmounted when you run `fsck`; the system must be in single-user mode when running `fsck` on the *root* filesystem. If the filesystem is corrupted to a point where `fsck` is unable to recover it, you may be able to fix the filesystem manually with `fsdb(ADM)`. See "Repairing a filesystem when `fsck` stops at size check" (page 912) for information. Otherwise, you may need to recreate the filesystem and restore the data from a backup.

Filesystems should always be checked after a system crash before they are mounted. Minor filesystem corruption can become widespread quickly if you continue to read and write a corrupted filesystem. `fsck` can be run with the `-y` option so that it will automatically attempt to fix any filesystem corruption it finds. `fsck -y` runs when rebooting the system if the appropriate string is in the files that control the booting process are set up appropriately:

Filesystem	String	File
root	FSCKFIX = YES	/etc/default/boot
all others	fsckflags = y	/etc/default/filesys

Without these lines, `fsck` reports each error it finds and waits for you to respond before it attempts to fix it during reboot. This prevents the automatic reboot feature from working if any filesystems suffered any damage at all from the crash.

be replaced by restoring them from backups. Permission and some "type" problems can be fixed with the `fixmog` utility. All errors found during the integrity check are packaged as audit records that show the audit event as a Database Event in the audit trail.

NOTE Some files may be listed as missing in a correctly configured system, such as one of the pair `/usr/lib/cron/at.allow` and `/usr/lib/cron/at.deny`.

ps(C) — check process activity

The `ps(C)` command obtains information about active processes. This command gives a "snapshot" picture of what processes are executing, which is useful when you are trying to identify what processes are loading the system. Things will probably change by the time the output appears; however, the entries that you should be interested in are **TIME** (minutes and seconds of CPU time used by processes) and **STIME** (time when the process first started). The following example contains sample output from the `ps -af` command.

Sample output from `ps -af`

```

UID    PID  PPID  C   STIME TTY   TIME COMMAND
root   22247 21299  0 16:54:41 T3    0:01 msctest -n 2
johnson 22246 22079  0 16:52:53 T0    0:04 vi file2
root   22285 22247  0 16:56:04 T1    0:06 msctest -n 2
root   22284 22247  0 16:56:04 T1    0:09 msctest -n 2
markham 22274 22271  0 16:55:09 p1    1:05 rlogin colossus
markham 22271 22243  0 16:55:07 p1    1:05 rlogin colossus
forbin 22304 21003  0 17:02:51 003   0:00 /usr/bin/mail kuptin
fisher 22299 18505  2 17:02:29 011   3:28 nerhack
root   22305 1327 10 17:02:57 T2    0:00 ps -af
```

NOTE If you booted your system from a file other than `/unix` (such as `/unix.old`), you must specify the name of that file with the `-n` option to `ps`. For example, `ps -ef -n unix.old`.

See "Runaway processes, stopping" (page 936) for information about using the `ps` command to perform such tasks as locating "runaway" processes (one that uses progressively more system resources over a period of time while you are monitoring it). You can also use `ps` to find processes that take a very long time to execute; you can consider using `cron(C)` to execute such jobs during an idle time of the day.

sar(ADM) — system activity reporter

sar(ADM) provides information that can help you understand how system resources are being used on your system. This information can help you solve and avoid serious performance problems on your system.

The following table summarizes the sar options:

sar options

Option	Reports on:
-a	file access system routines
-b	buffer cache activity
-c	system calls
-d	disk activity
-m	I/P message and semaphore activity
-n	name cache
-p	paging activity
-q	average length of run and swap queues
-r	unused memory pages and disk blocks
-u	CPU utilization
-v	process, inode, file, and lock table status
-w	system swapping and switching activity
-y	TTY device activity
-A	summary of all reports

The individual sar outputs are discussed in alphabetical order later in this chapter.

How sar works

Internal activity is measured by a number of counters contained in the kernel. Each time an operation is performed, an associated counter is incremented. The sar(ADM) utility generates reports based on the raw data gathered from these counters. sar reports can be used to diagnose system problems. The two most critical areas to monitor are memory and CPU usage. The functions monitored by sar are discussed in the subsections that follow, including analysis of sample sar output. sar can either gather system activity data "live" or extract information collected in data files created by sadc (System Activity Data Collector).

By default, the following **crontab** entry is installed in **sys** file in the **/usr/spool/cron/crontabs** directory:

```
0 * * * 0 6 /usr/lib/sa/sa1
20,40 8-17 * * 1-5 /usr/lib/sa/sa1
5 18 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 19:01 -i 1200 -A
```

The **sa1** entries produce records every 20 minutes during working hours and hourly otherwise. The **sa2** entry writes a daily average report into the same file every hour during work hours. You can modify these entries as desired. The output files are in binary format (for compactness) and are stored in **/usr/adm/sa**. The filenames have the format **sadd**, where **dd** is the day of the month. (See the **crontab(C)** and **sar(ADM)** manual pages for more information on modifying the default **crontab** file.)

Running *sar*

The basic syntax of the **sar(ADM)** command is as follows:

```
sar [-option] [file]
```

Here

option one of the options described in the following sections

file name of the file for the day whose statistics you want to view. For example, to view the **sar -v** report for the tenth day of the most recent month, the command is:

```
sar -v -f /usr/adm/sa/sa10
```

You can also run **sar** in "real time". To do this, specify the sampling interval in seconds. For example, to take a sample every 15 seconds, the command is:

```
sar -v 15
```

The released system allows any user to run **sar** in real time. However, the files in the **/usr/adm/sa** directory are readable only by **root**. Change the permissions on the files in that directory to allow other users to access the **sar** data.

The examples in this section are not intended to represent benchmarks; they serve to illustrate how the output can be used. When tuning your system, it is recommended that you use a "benchmark" (a program used to evaluate the performance of a system) and have the system under normal load for your application.

sar -a — file access operations

The `sar -a` option reports the use of file access operations. The operating system routines reported are as follows:

- `iget/s` number of files located by i-node entry per second
- `namei/s` number of filesystem path searches per second. "namei" calls "iget", so "iget/s" is always larger than "namei/s".
- `dirbk/s` number of directory block reads issued per second

An example of `sar -a` output, with a 30-second sampling interval, follows:

```

unix snlx 3.2 2 1386 07/18/92
12:41:18  iget/s namei/s dirbk/s
12:42:18      4         1         3
12:42:48      2         1         1
12:43:18      5         2         3
Average      4         1         3

```

The larger the values reported, the more time the kernel spends accessing user files. This indicates how heavily programs and applications are using the filesystem(s). In general, if the ratio of "iget/s" to "namei/s" is greater than 5 and "namei/s" is greater than 30, it may indicate that your filesystem organization is inefficient. Refer to "Checking free space on filesystems" (page 911) and "Reducing disk fragmentation" (page 911) for ways to solve the problem.

sar -b — *buffer activity*

The **-b** option reports the following buffer activity.

bread/s	average number of physical blocks read into the system buffers from the disk (or other block devices) per second
lread/s	average number of logical blocks read from system buffers per second
%rcache	fraction of logical reads found in buffer cache (100% minus the ratio of breads to lreads)
bwrit/s	average number of physical blocks written from the system buffers to disk (or other block devices) per second
lwrit/s	average number of logical blocks written to system buffers per second
%wcache	fraction of logical writes found in buffer cache (100% minus the ratio of bwrit/s to lwrit/s)
pread/s	average number of physical read requests per second
pwrit/s	average number of physical write requests per second

The entries that you should be most interested in are the cache hit ratios “%rcache” and “%wcache”, which measure the effectiveness of system buffering. If “%rcache” falls below 90, or “%wcache” falls below 65, it may be possible to improve performance by increasing the number of buffers. If your application is I/O-intensive and you have a large memory configuration, you may want to tune the buffer cache so that %rcache is around 95 and %wcache is around 80. The number of buffers in the system buffer cache are controlled by the NBUF tunable parameter. See “Tuning I/O parameters” (page 843) for more information.

Technically, 100% hit ratio for the read cache should mean better performance than the 90% recommended here, but getting that last 10% of hits often requires so much memory that performance suffers because of the lack of user memory for executing processes.

An example of **sar -b** output follows:

```
unix unix 3.2 2 i386 07/18/92
16:32:57 bread/s lread/s %rcache bwrit/s lwrit/s %wcache pread/s pwrit/s
16:33:07 3 39 93 1 15 91 0 0
16:33:17 4 40 90 2 16 87 0 0
16:33:27 4 41 90 3 17 84 0 0
Average 4 40 91 2 13 84 0 0
```

This example shows that the buffers are not causing any bottlenecks, because all data is within acceptable limits.

sar -b — *buffer activity*

The *-b* option reports the following buffer activity.

<i>bread/s</i>	average number of physical blocks read into the system buffers from the disk (or other block devices) per second
<i>lread/s</i>	average number of logical blocks read from system buffers per second
<i>%rcache</i>	fraction of logical reads found in buffer cache (100% minus the ratio of breads to lreads)
<i>bwrit/s</i>	average number of physical blocks written from the system buffers to disk (or other block devices) per second
<i>lwrit/s</i>	average number of logical blocks written to system buffers per second
<i>%wcache</i>	fraction of logical writes found in buffer cache (100% minus the ratio of bwrit/s to lwrit/s)
<i>pread/s</i>	average number of physical read requests per second
<i>pwrit/s</i>	average number of physical write requests per second

The entries that you should be most interested in are the cache hit ratios "*%rcache*" and "*%wcache*", which measure the effectiveness of system buffering. If "*%rcache*" falls below 90, or "*%wcache*" falls below 65, it may be possible to improve performance by increasing the number of buffers. If your application is I/O-intensive and you have a large memory configuration, you may want to tune the buffer cache so that *%rcache* is around 95 and *%wcache* is around 80. The number of buffers in the system buffer cache are controlled by the *NBUF* tunable parameter. See "Tuning I/O parameters" (page 843) for more information.

Technically, 100% hit ratio for the read cache should mean better performance than the 90% recommended here, but getting that last 10% of hits often requires so much memory that performance suffers because of the lack of user memory for executing processes.

An example of *sar -b* output follows:

```
unix unix 3.2 2 i386 07/18/92
16:32:57 bread/s lread/s %rcache bwrit/s lwrit/s %wcache pread/s pwrit/s
16:33:07 3 39 93 1 15 91 0 0
16:33:17 4 40 90 2 16 87 0 0
16:33:27 4 41 90 3 17 84 0 0
Average 4 40 91 2 13 84 0 0
```

This example shows that the buffers are not causing any bottlenecks, because all data is within acceptable limits.

sar -c — system call activity

The `-c` option reports system calls in the following categories:

scall/s	all types of system calls per second, generally about 30 per second on a busy 4 to 6 user system
sread/s	read system calls per second
swrit/s	write system calls per second
fork/s	fork system calls per second, about 0.5 per second on a 4 to 6 user system. This number increases if shell scripts are running.
exec/s	exec system calls per second. If (exec/s) / (fork/s) is greater than 3, look for inefficient \$PATH's)
rchar/s	characters (bytes) transferred by read system calls per second
wchar/s	characters (bytes) transferred by write system calls per second

Typically, reads plus writes account for about half of the total system calls, although this varies greatly with the activities that are being performed by the system.

An example of `sar -c` output follows:

```

unix unix 3.2 2 1986 07 1-1987
18:33:04 scall/s sread/s swrit/s fork/s exec/s rchar/s wchar/s
18:33:35      38      16      5      0.03      0.07      608      1000
18:34:05      39      17      4      0.07      0.07      612      1000
18:34:35      38      17      5      0.07      0.07      604      1000
Average      38      16      5      0.03      0.07      608      1000

```

This report is of interest mostly for programmers who are testing application programs. If `scall/s` is greater than 300 over an extended period of time, it may indicate inefficient code for applications that are running.

Note that the XENIX system `vmstat(C)` command that is also supported on Open Desktop provides cumulative statistics about system calls executed (`vmstat -s`) and number of forks (`vmstat -f`).

sar -m — interprocess communication

The **sar -m** option reports on interprocess communication activities. Message and semaphore calls are reported as follows:

msg/s number of message operations (sends and receives) per second
 sema/s number of semaphore operations per second

An example of **sar -m** output follows:

```

unix unix 3.2 2 1386 02/18/92
15:16:58 msg/s sema/s
15:17:32 12.3 29.2
15:18:02 11.8 26.9
15:18:32 18.4 23.1
Average 14.2 26.4

```

If you are not running application programs that use messages and System V semaphores, these figures will all be equal to 0.00. If you are using these interprocess communications facilities and either msg/s or sem/s is greater than 100, the application is not using the system efficiently. See "Message queues" (page 883) and "Semaphores" (page 884) for information about tuning these parameters.

sar -n — name cache activity

The **sar -n** option reports the name cache statistics.

c_hits number of cache hits
 cmisses number of cache misses
 hit% hit to miss ratio as a percentage

An example of **sar -n** follows:

```

unix unix 3.2 2 1386 02/18/92
11:26:05 c_hits cmisses (hit %)
11:26:10 9 0 (100%)

```

The **name1** cache improves the time required to search a full pathname when first accessing a file. Generally, the higher the "hit %" field the better. If the "hit %" consistently falls below 90%, increase the value of **S5CACHEENTS** and associated parameters. See "Tuning I/O parameters" (page 843) and **S5CACHEENTS** (page 879) for more information.

sar -p — paging activity

The **-p** option reports paging activity. The following page rates are recorded.

vflt/s	number of address translation page faults per second (valid page not present in memory)
pflt/s	number of page faults from protection errors per second (illegal access to page) or "copy-on-writes", pflt/s generally consists entirely of "copy-on-writes"
pgfil/s	number of vflt/s per second satisfied by a page-in from the file-system (each pgfil causes two lreads; see sar -b)
rcfm/s	number of valid pages per second that the system has reclaimed (added to list of free pages)

An example of **sar -p** output follows:

```

unix unix 3.2 2 1386    02-18-92
12:01:51 vflt/s pflt/s pgfil/s rcfm/s
12:56:52 13.91  2.30  5.63  11.11

```

High values (over 100) for vflt/s can indicate that application programs are not efficient for a paging system (poor locality of reference), paging tunables need adjustment, or the memory configuration is inadequate for the system load.

The **vmstat(C)** command also reports statistics about paging activity. An example of the information provided by the **vmstat -s** command is:

```

1100 free swap space
12002 demand zero and demand full pages
25932 pages on swap
44569 pages in cache
28719 pages on file
33791 protection fault
84644 pages are freed
  23 success in swapping out a process
   0 fail in swapping in a process
  22 success in swapping in a process
  98 swapping out a region
  64 swapping in a region
457461 cpu context switches
1870524 system calls

```

This output is representative of a system where excessive paging may be degrading performance.

sar -q — process throughput

The `sar -q` option reports the average queue length while the queue is occupied and percent of time occupied.

`runq-sz` run queue of processes ready to run
`%runocc` percentage of time the run queue is occupied
`swpq-sz` swap queue of processes to be swapped out
`%swpocc` percentage of time the swap queue is occupied

An example of `sar -q` follows:

```

unix unix 3.2 2 i386 02/18/92
11:00:56 runq-sz %runocc swpq-sz %swpocc
11:01:07 1.7 98 1.5 36
11:01:17 1.0 63 1.0 31
11:01:27 1.0 58 1.0 49
Average 1.3 74 1.2 39

```

If “`%runocc`” is greater than 90 and “`runq-sz`” is greater than 2, the CPU is heavily loaded and response time may be degraded. In this case, moving to a multiprocessing configuration might improve system response time. If “`%swpocc`” is greater than 20, more memory or fewer buffers would help reduce swapping/paging activity.

sar -r — page and swap usage

The `-r` option records the number of memory pages and swap file disk blocks that are currently unused. The following are recorded.

`freemem` average number of 4K pages of memory available to user processes over the intervals sampled by the command
`freeswap` number of 512-byte disk blocks available for process swapping

An example of `sar -r` output follows:

```

unix unix 3.2 2 i386 02/18/92
12:01:51 freemem freeswap
12:56:52 208 5848

```

sar -u — CPU utilization

The CPU utilization is listed by `sar -u` (or by the `sar` command with no options specified). At any given moment the processor is either busy or idle. When busy, the processor is in either user or system mode. When idle, the processor is waiting for input/output completion or has no work to do. The `-u` option of `sar` lists the percent of time that the processor is in system mode (`%sys`), user mode (`%user`), waiting for input/output completion (`%wio`), and idle time (`%idle`). An example of `sar -u` follows:

Time	%usr	%sys	%wio	%idle
10:20:05	6	7	2	85
10:40:12	6	7	2	85
10:00:03	7	9	3	81
10:20:07	14	16	10	60
Average	9	11	5	75

In typical timesharing use, `%sys` and `%usr` are about the same value. In special applications, either of these can be larger than the other without anything being abnormal. A high `%wio` generally means a disk bottleneck. A high `%idle`, with degraded response time, may mean memory constraints; time spent waiting for memory is attributed to `%idle`.

The "idle" (percentage idle) column can also provide some insight into system performance. This figure is normally between 40 and 100 percent, even with a large number of active users. When this figure falls consistently below 30%, the chief competition for resources does not involve memory at all; the critical resource is raw processor power. (Run the `ps(C)` command to make certain that the excessive CPU usage is not due to a runaway process that is stealing every spare CPU cycle.)

If you are running a large number of users, it may help to switch to intelligent serial boards if you are using non-intelligent cards. Intelligent cards take some of the burden off the CPU rather than adding to the amount of work it has to do.

In addition, you should examine the files in `/usr/spool/cron/crontabs` to see if jobs are queued up for peak periods that might better be run at times when the system is idle. Use the `ps` command to determine what processes are heavily loading the system. Encourage users to run large, non-interactive commands at off-peak hours. You may also want to run such commands with a low priority by using the `nice(C)` or `batch(C)` commands.

sar -v — system tables

The **-v** option reports the status of selected kernel tables. From this report you know when the size of these system tables need to be modified.

- proc-sz** number of process table entries presently being used/allocated in the kernel (controlled by **NPROC**)
- inod-sz** number of inode table entries presently being used/allocated in the kernel (controlled by **NINODE**)
- file-sz** number of file table entries presently being used/allocated in the kernel (controlled by **NFILE**)
- ov** number of times an overflow occurred (one column for each of the above three items)
- lock-sz** number of record lock table entries presently being used/allocated in the kernel (controlled by **FLCKREC**)

The values are given as level/table size, meaning the current number of table entries in use and the size of the table. An example of **sar -v** follows:

```

unix unix 3.2 2 i386 02/18/92
17:36:05 proc-sz ov inod-sz ov file-sz ov lock-sz
17:36:35 247 350 0 1737 1050 0 1537 1400 0 37 100
17:37:05 267 350 0 1927 1050 0 1637 1400 0 37 100
17:37:35 272 350 0 1457 1050 0 1407 1400 0 37 100

```

This example shows that all tables are large enough to have no overflows. Sizes could be reduced to save main memory space if these are the highest values ever recorded.

To see the actual number of these table entries being used currently, use the following command:

```
sar -v 1 1
```

The **pstat(C)** command provides similar information in a different format. To see the actual number of these table entries being used currently, you can also use this command:

```
pstat | grep '[0-9]'
```

sar -w — swapping activity

The **-w** option reports swapping and context-switching activity. The following statistics are reported:

- swpin/s** number of transfers into memory per second
- bswin/s** number of 512-byte-block units (blocks) transferred for swap-ins (including initial loading of some programs) per second
- swpot/s** number of transfers from memory to the disk swap area per second. If greater than 1, memory may need to be increased or buffers decreased.
- bswot/s** number of blocks transferred for swap-outs per second
- pswch/s** process switches per second. This should be 30 to 50 on a busy 4 to 6 user system.

An example of **sar -w** output follows:

```

unix unix 3.2 2 1386 02/18/92
19:53:44 swpin/s bswin/s swpot/s bswot/s pswch/s
19:53:58 0.0 0.0 0.0 0.0 37
19:54:14 0.0 0.0 0.0 0.0 37
19:54:24 0.0 0.0 0.0 0.0 37
Average 0.0 0.0 0.0 0.0 37

```

This example shows that there is sufficient memory for the currently active users, because no swapping is occurring. Many system administrators aim for zero swapping, which can be accomplished by adding physical memory. Disk accesses are significantly slower than memory accesses, so minimizing swapping activity is important for good performance.

The **vmstat(C)** command also provides information about swapping and context-switching activity. Use **vmstat -s** to view this data.

sar -y — terminal I/O

The **-y** option monitors terminal device activities. If you have a lot of terminal I/O, you can use this report to determine if there are any bad lines. Activities recorded are defined as follows:

rawch/s input characters (raw queue) per second

canch/s input characters processed by canon (canonical queue) per second. Canonical processing recognizes interrupt characters ((Del) by default), quit (<\), suspend ((Ctrl)Z), and editing characters such as (Bksp) and kill ((Ctrl)U). For example, if the user types:

ha(Bksp)ello

it increments the value of "rawch/s" by 7 and the value of "canch/s" by 5.

outch/s output characters (output queue) per second

rcvin/s receiver hardware interrupts per second

xmtin/s transmitter hardware interrupts per second

mdmin/s modem interrupts per second

Not all terminal drivers are written to produce these statistics. All SCO serial, console, and pseudo-tty drivers and most drivers produced by other vendors produce the rawch/s, canch/s, and outch/s statistics.

The number of modem interrupts per second (mdmin/s) should be close to 0. The ratio of xmtin/s to rawch/s and canch/s, and the ratio of rcvin/s to outch/s should be fairly constant. For non-intelligent I/O cards, these ratios should be close to 1:1. Intelligent I/O drivers move many characters per interrupt, and some drivers do not even use interrupts, so the ratios will be different. If the number of interrupts per transmitted character starts to increase dramatically, it usually indicates a bad line that is generating extraneous interrupts.

An example of sar -y output follows:

```

unix unix 3.2 2 1386    02/18/92
16:50:11 rawch/s canch/s outch/s rcvin/s xmtin/s mdmin/s
16:50:41      112      15      653      103      102      0
16:51:11      107       7      654      104      105      0
16:51:41       99       5      641       99      105      0
Average      106       9      649      102      104      0

```

swap(ADM) — check and add swap space

Swap space is disk storage that is part of virtual memory. When main memory is exhausted, pages of processes that are sleeping awaiting their turn to execute can be written out to a swap device, thus freeing up main memory for executing processes. The system comes configured with one swap device. Adding additional swap devices with the `swap(ADM)` command makes more memory available to user processes. Excessive paging degrades system performance, but augmenting the swap space is a way to make more memory available to executing processes without tuning the kernel.

The following command adds a second swap device, `/dev/swap1`, to the system. The swap area starts 0 blocks into the swap device and the swap device is 16000 512-byte blocks in size.

```
swap -a /dev/swap1 - 16000
```

Use the `swap -l` command to see statistics about all the swap devices currently configured on the system.

Running the `swap -a` command adds a second swap device only until the system is rebooted. To ensure that the second swap device is available every time the system is rebooted, use a startup script in the `/etc/rc2.d` directory. For example you could call it `S09AddSwap`.

timex(ADM) — examine system activity per command

The `timex` command times a command and reports the system activities that occurred on behalf of the command as it executed. Run without options, `timex` reports the amount of real (clock) time that expired while the command was executing and the amount of CPU time (user and system) that was devoted to the process. For example:

```
# timex command command_options
real    0:54.35
user    53.98
sys     14.86
```

Running `timex -s` is roughly equivalent to running `sar -A`, but it displays system statistics only from when you issued the command until the command finished executing. If no other programs are running, this information can help identify which resources a specific command uses during its execution. System consumption can be collected for each application program and used for tuning the heavily loaded resources.

Other information is available if the process accounting software is installed; see the `timex(ADM)` manual page for information. To enable process accounting, remove the comment characters from the command lines in the `/etc/rc.d/6/kinit` file and reboot the system.

tunesh(ADM) — automatic configuration tool

`tunesh` is an automatic configuration utility supplied with Open Desktop. It adjusts the values of key tunable parameters based on information you supply about your system. You should run `tunesh` soon after installing the system to determine initial values for these tunable parameters; as you add memory, terminals, filesystems, and so forth, you can run the configuration program again to modify system parameters accordingly.

WARNING `tunesh` sets system resources to levels appropriate for your system configuration, but may not result in optimum system performance. Study the `sar` reports and heed messages on the console about insufficient resources to understand the usage patterns for your particular system; in many cases, you may want to do additional tuning with the `configure(ADM)` command to allocate resources to meet your usage patterns.

`tunesh` bases the configuration on the information you supply about the following:

- amount of memory (RAM) installed (detected automatically)
- number of filesystems exported and imported via NFS
- number of local filesystems that will be mounted

On the server system, you are also asked about the following:

- number of X terminals
- number of serial login sessions (serial terminals and modems)
- number of remote login sessions (via `rlogin`, `telnet`, and `ftp`)

When running `tunesh` on the Personal System, the system makes calculations that assume 2 serial login sessions, 2 X-terminals, and 2 remote logins, to match the limits imposed by the Personal System license.

3. Tools reference

To reconfigure the system with **tunesh**:

1. Log in as *root* and issue the `/usr/lib/tunesh/tunesh` command.
2. Enter the new values as appropriate.
3. Relink the kernel, following the prompts.
4. When the program exits, shut the system down and reboot. If you are doing this during working hours, you should warn your users that the system will be unavailable for 10-15 minutes. To shut down the system, make the following `sysadmsh` selection:

System ↪ Terminate

If the system is under use, give people several minutes to finish their work and log out.

You can run the program again any time your configuration changes. Every time `tunesh` is used to change the system configuration, a mail message is sent to the *root* account with the heading "ODT System configuration log." This message contains a list of parameters changed and their new values, plus any error messages encountered during the tuning process. This mail serves as a record of the changes made to the system and should be filed with your System Log.

XENIX System tools

The `pstat(C)` and `vmstat(C)` commands that were originally developed for the XENIX operating system are also supported on Open Desktop. These commands provide much of the same information available through the `sar(ADM)` and `crash(ADM)` commands, but in a different format. See the manual pages for more information.

INSTALACIÓN Y MANEJO DE REDES (LAN) EN PLATAFORMA UNIX

7.- INTEGRACIÓN DE UNIX CON OTROS SISTEMAS



Mayo de 1996.

Chapter 14

Getting started with DOS

In all important respects, using Open Desktop DOS Services is just like using DOS on a stand-alone personal computer. With Open Desktop, you can:

- use all common DOS commands.
- install and run off-the-shelf DOS applications.
- use your computer hardware (diskette drives and printers, for example) in standard DOS ways.

In addition, with DOS Services, you can:

- run several DOS applications in separate DOS environments simultaneously and switch between any of the DOS and UNIX windows.
- take advantage of the security capabilities of the UNIX system, including password protection for the whole system and protection for specified DOS directories, data files, and programs.
- access UNIX data files and programs, including files within a network environment.

This is possible because DOS Services creates a *virtual personal computer* (also called a *virtual PC* or *virtual machine*) for you whenever you run DOS. Because DOS Services virtual computers use the virtual 8086 mode of your 80386 or 80486 processor, you can run DOS commands and applications under DOS Services only if they are compatible with the Intel 8086 processor. DOS Services does not support DOS applications that require the protected mode available on 80286, 80386, and 80486 processors.

Beginning a DOS session

You can enter the DOS environment from the Desktop in one of three ways. You can:

1. Double-click on the DOS icon from the Desktop. (The DOS icon is located in the Accessories window, which can be opened by double-clicking on the Accessories icon.)

-or-

2. Double-click on the UNIX icon to open a UNIX window, and then type `dos` at the UNIX prompt.

-or-

3. Double-click on an icon that represents a DOS program (an executable DOS file).

If you use either method 1 or method 2, the DOS window opens with the the standard DOS prompt displayed:

```
C:
```

If you use method 3, the DOS window opens with the the selected program already running. For example, double-clicking on the Lotus 1-2-3 icon causes Lotus 1-2-3 to run without first displaying a `C:` prompt.

The `C:` prompt that you see after invoking DOS with either of the first two methods tells you that you are using DOS drive C: (the fixed disk). You can now use your computer as you would use a standard computer running DOS.

See "Controlling the DOS window" (page 146) for more about the DOS window.

Ending a DOS session

To end a DOS session and return to the Desktop, type (at your DOS prompt):

```
C: quit
```

Or, if you are running a DOS executable file (such as Lotus 1-2-3), simply exit the program. The Desktop reappears, and you can continue to use Open Desktop.

Using DOS commands and applications

All common DOS commands work as they do on a conventional, stand-alone DOS computer.

You can use DOS commands for routine operations like copying files or listing the names of files on the fixed disk or a diskette. There are also DOS commands for more specialized purposes such as creating text files and creating and executing BASIC programs. These DOS commands are all supplied with DOS Services and are described in your DOS documentation. You can also install and use off-the-shelf DOS applications in the DOS environment. *Administering DOS Services* in the *System Administrator's Guide* provides instructions for installing DOS applications for use on Open Desktop.

In the DOS environment, you specify directories and give options to commands in the usual DOS way. The following command displays the contents of the directory `\USR\DBIN` in wide format, with five files listed per line:

```
C: dir \usr\dbin /w
```

If you are a UNIX user who has not used DOS, you may be unfamiliar with the use of the slash (/) to turn on options and the backslash (\) as the path separator. For further information on this syntax, refer to your DOS documentation.

Changing the default drive: When you enter the DOS environment, your default drive is drive C: (the fixed disk) and your prompt is C:>. To change your default drive to drive A:, be sure you have a valid, formatted DOS diskette in the drive, and type:

```
C: a:
```

Your prompt changes to A> and you can execute commands from the diskette drive.

If your system has a second diskette drive, you can use it with DOS by referring to it as drive B:. The diskette drives are available on a first-come-first-served basis. If one user is accessing a diskette drive and a second user attempts to use it at the same time, the second user sees a message stating that the drive is unavailable.

Changing directories: Use the DOS `CD` or `CHDIR` command to change your current working directory. To change to `\USR\DBIN`, for example, type:

```
C: cd \usr\dbin
```

Piping and redirecting with DOS: Pipes and redirection function in DOS Services as under standard DOS:

```
C: dir | sort > contents
C: dir a: >> contents
```

All common DOS commands work as you would expect in the DOS Services environment, including COPY, for copying files; COMP, for comparing files; TYPE, for displaying the contents of files; REN, for renaming files; and DEL, for deleting files.

The more specialized DOS tools for editing files, programming, and configuring the DOS environment also work in the DOS Services environment. These tools include:

- the EDLIN and EDIT editors, and the QBASIC interpreter,
- batch files, including all standard batch commands,
- the DEBUG utility, and
- CONFIG.SYS files.

Using off-the-shelf DOS applications

You can use most off-the-shelf DOS application programs in the DOS Services environment just as you would use them on a stand-alone DOS personal computer. You can also use custom DOS applications that you might have developed.

To use an application from drive A:, follow the application manufacturer's instructions. Typically, you insert the application diskette into drive A:, change your current drive to drive A:, and invoke the application by name from your A> prompt. For example:

```
C: a:
A: wp
```

To run an application from drive C:, you must first install the application on the fixed disk.¹ Once installed, applications are executed according to the manufacturer's instructions. For example, if WordPerfect is installed on your fixed disk, you can start it by typing:

```
C: wp
```

1. In most cases, DOS applications are installed by following the application manufacturer's instructions. For further pointers on installing DOS applications, see *Administering DOS Services* in the Open Desktop Administrator's Guide.

Booting applications from drive A:

A few personal computer applications (such as some versions of the Microsoft Flight Simulator[®]) must be booted from drive A: because they do not run under DOS. (They actually boot their own operating systems.) To use these applications on a conventional personal computer, you insert the bootable application diskette into drive A: and power the computer on or press <Ctrl><Alt>.

To run these applications on DOS Services, you use the `dosboot` command. To use `dosboot`, you must be using the UNIX shell and not the DOS environment. If you are currently in the DOS environment, type `quit`. Your prompt should be `$` or `%`.

Only two steps are required to use `dosboot`:

1. Insert your bootable application diskette into drive A: and lock it in place.
2. Type:

```
$ dosboot
```

When you use `dosboot`, your application runs independently of any other UNIX or DOS activity. This means that files on drive C: are not available, and you cannot type `quit` as you usually do to leave the DOS environment.

To end a `dosboot` session, press <Ctrl><Alt>.² Your UNIX system prompt then returns.

File permission errors

Sometimes the message DOS returns is affected by file permission modes. For example, when a DOS command you issue encounters a file for which you do not have read access, DOS may display a message that implies the file does not exist, even though the file does exist. Similarly, if you try to create a file in a directory for which you do not have write access, DOS may display an error message such as `File creation error` that does not clearly indicate the nature of the problem.

2. Note that you can also use the KILL DOS control code described in "Stopping DOS programs" (page 129) if your application is hung and does not respond to <Ctrl><Alt>.

Inapplicable DOS commands

Nearly all standard DOS commands operate in the DOS Services environment just as they do on a conventional stand-alone DOS computer. Some DOS commands, however, are either not useable in the DOS Services environment or operate differently than they do on a stand-alone DOS computer.

In particular, some of them will operate correctly only on a "real" DOS filesystem. DOS filesystems are indexed by a File Allocation Table. DOS Services emulates DOS filesystems while preserving the underlying UNIX structure, which is completely different.

The following restrictions apply:

- You cannot use the DOS **FDISK** command under DOS Services. Instead of running **FDISK** under DOS Services, use equivalent UNIX utilities or shut down the UNIX system, boot standard DOS, and use **FDISK** under standard DOS.
- You cannot use **SHIP** or any other DOS command for parking the fixed disk head on the DOS Services system.
- You cannot use the following commands on the shared UNIX/DOS filesystem: **CHKDSK**, **FORMAT**, **SYS**, **MIRROR**, or **UNFORMAT**. Do not use them on Drives C:, D: or J:.

You *can* use these commands on a real DOS filesystem, such as the diskette drive, or a physical DOS partition. You can also use them on virtual floppies and virtual DOS partitions, because, though these are portions of the shared UNIX/DOS filesystem, they are formatted as real DOS filesystems.

FORMAT may work somewhat differently under DOS Services and standard DOS. A filesystem which you **FORMAT** under DOS Services may work properly under DOS Services but not work properly under raw DOS. It is safest to use raw DOS to **FORMAT** any disk or partition from which you intend to boot DOS. They will then work properly under either DOS or DOS Services.

Since virtual floppies or virtual partitions will be used only under DOS Services, they should be **FORMATED** or **UNFORMATED** under DOS Services.

- Similarly, you cannot use **UNDELETE** on any file which is part of the shared UNIX/DOS filesystem. But you *can* use it on any file which is part of a real DOS filesystem. Use **UNDELETE** on real or virtual floppies or on real or virtual DOS partitions.

- You *can* use the DOS TIME and DATE commands to display or change the time and date that apply to the DOS environment, but when you leave the DOS environment, time and date are determined by the UNIX clock. When you reenter DOS, the DOS clock is always initially synchronized with the UNIX clock.

If you issue a DOS command that does not work in the DOS Services environment, DOS displays an error message but does not harm your computer in any way or destroy any data.

Stopping DOS programs

There are several ways to stop DOS programs that you start in a DOS environment. Most DOS applications include a specific procedure for stopping their execution. Whenever possible, you should stop a DOS program using the procedure designed for that program. Sometimes, however, you might want to stop a DOS utility that provides no specific method for termination, or else a DOS application might get locked into a state where the prescribed termination procedure does not work. If you run into one of those conditions, follow one of these procedures:

1. Use the DOS break character `<Ctrl>C` or `<Ctrl>Break` just as you would in standard DOS. These functions stop DOS commands like DIR, TYPE, or TREE, and some applications. When you press `<Ctrl>C`, your DOS prompt returns and you can resume DOS work immediately.
 2. If `<Ctrl>Break` does not work, press `<Ctrl>Alt>Del`. That is, press `<Ctrl>` and `<Alt>` at the same time; then, while still holding `<Ctrl>` and `<Alt>`, press ``. This is the key sequence used to reboot DOS on a standard DOS computer. In DOS Services, `<Ctrl>Alt>Del` causes the DOS program as well as the current DOS environment to abort. You must reinvoke DOS before you can resume DOS work.
- WARNING** When you press `<Ctrl>Alt>Del`, you could lose data if your DOS program is working on open files, just as you would on any standard DOS system.
3. If neither the `<Break>` character nor `<Ctrl>Alt>Del` properly terminates your DOS process, use the KILL DOS control code appropriate for your terminal.

DOS Window or Console	PC Scancode Terminal	ASCII Terminal
<code><Ctrl>Esc</code> <code><Ctrl>K</code>	<code><Ctrl>Esc</code> <code><Ctrl>K</code>	<code><Esc></code> <code><Ctrl>K</code>

Chapter 15

Finding your way around with DOS

With Open Desktop DOS Services the entire UNIX filesystem is available to you. DOS treats it as a DOS fixed disk, usually referenced as Drive C:.

When you boot DOS on a conventional stand-alone personal computer, your working directory is the root of the filesystem tree. You own all files in the filesystem and can access them easily with CD (change directory) commands. You can also modify any file as you please.

In Open Desktop, each user has a home directory, that is, a directory containing the files and subdirectories created and owned by that user. When you log in to the UNIX environment in Open Desktop and then immediately enter the DOS environment, your working directory is your home directory. (If you change directories before entering the DOS environment, however, your working DOS directory is the same as your UNIX directory at the time you type dos.) You can access your own files and subdirectories like you can on a conventional DOS computer.

The DOS search path

When you run a DOS program by typing a path name, DOS looks in the directory you specify for the program. If the program is there, DOS runs it. If the program is not there, the operation fails. For example, if you type:

```
C: \usr\ldbin\wp5\wp
```

DOS looks in the directory `\USR\LDBIN\WP5` for the program WP and runs it only if it is there.

If you type only the name of the program (for example, `wp`) without specifying its path, DOS looks first in your current working directory for the program. If the program is there, DOS runs it. If the program is not there, DOS searches through the directories in your search path to find the program.

The DOS search path in DOS Services works like the search path on a conventional DOS system, with one difference: when you enter the DOS environment, your search path is automatically set to be the same as your UNIX search path. This path includes the directories `\USR\DBIN` and `\USR\LDBIN`, the directories where standard DOS commands and applications are stored. You can, if necessary, override the default DOS search path by using the `PATH` command as you would on a conventional DOS system. Note that the path is often set in `AUTOEXEC.BAT`.

Naming DOS files and directories

When you create files or directories during a DOS session or using a DOS application, your names must conform to standard DOS rules for length and character set.

You can type the name with either uppercase or lowercase alphabetic characters. When you create a file on a DOS medium (a diskette in drive A:, for example), DOS converts all alphabetic characters to uppercase as expected. When you use DOS to create a file in the shared UNIX/DOS filesystem (DOS drive C:), however, DOS Services converts all alphabetic characters to lowercase. Using lowercase for filenames is conventional under the UNIX system. Requiring names to be consistently lowercase also prevents you from creating names that are identical except for case, which DOS cannot differentiate.

Thus, any file you create with DOS Services in the shared UNIX/DOS filesystem can be accessed by either DOS or the UNIX system.

Differences between DOS and UNIX filenames

DOS and UNIX rules for naming files and directories differ with respect to case, size, and character set.

UNIX is case-sensitive while DOS is not. Alphabetic characters in UNIX file and directory names are usually lowercase, but they can be any combination of upper- and lowercase. Whatever combination you enter is preserved. DOS, on the other hand, interprets all alphabetic characters in file- and directory names as uppercase, whether you enter them in uppercase or lowercase. To the UNIX system, "chapter1" and "CHAPTER1" name two different files; DOS cannot distinguish between the two forms, seeing them instead as the same name.

DOS limits file and directory names to eight characters plus an optional extension of up to three characters. Traditionally, the UNIX system allows names up to 14 characters, although some newer systems (including Open Desktop 2.0) allow more. Although the UNIX system does not provide for filename extensions in the same sense as DOS, a UNIX name can contain a period anywhere in the name. Thus, while the UNIX system accepts any legal DOS name, DOS does not allow such perfectly good UNIX names as *messagetoall* or *chapter.seven*.

DOS and the UNIX system accept both alphabetic and nonalphabetic characters in file and directory names, but the UNIX system accepts more nonalphabetic characters than DOS. For example, control characters and spaces are valid characters in UNIX names but not in DOS names. (Note that UNIX names containing spaces must be enclosed in quotes.)

Accessing files with illegal DOS names

You can use DOS to access any file or directory in the shared UNIX/DOS file-system, whether it was created with DOS or under the UNIX system. However, you must use a special mapped name for UNIX files or directories with names that do not conform to DOS rules. These names include:

- names longer than DOS allows.
- names with more than three characters following a period.
- names with nonalphabetic characters that DOS does not recognize.
- names with uppercase alphabetic characters.

When any DOS utility or application accesses a UNIX name that does not conform to DOS rules, DOS Services translates, or maps, the name to a legal DOS name by appending a unique index consisting of an apostrophe followed by one to three characters. If necessary, the UNIX filename is truncated before appending the index. For example, a file called *messagetoall* might be mapped to the name *mess'baq*. You can determine the mapped name by issuing the DOS DIR command.

Use the mapped names shown in the directory listing whenever you need to refer to UNIX files in a DOS command.

DOS

Examples of mapped filenames

The following table illustrates the operation of DOS Services filename-mapping on various types of UNIX filenames. The UNIX name is shown in the left column. A typical mapped name is shown in the right column.

UNIX name	Mapped name
Mail	MAIL\FPE
messagetoall	MESS\BAQ
message.tobob	MESS\BBF.TOB
+ .toomuch	_PS.TOO
:rofix	_ROF\CBL
:rofix.xtn	_ROF\BPQ.XTN
=.ok	_PP.OK
MiXcAsE.xtn	MIXCA\SU.XTN
okbase.:=+	OKBAS'QW.---
a.b.c	A_B'SV.C

Note that you need to use mapped filenames only when you use files created under the UNIX system with names that are not legal DOS names.

Displaying UNIX-style directory listings

Although you always use a UNIX file's mapped name with DOS commands, you sometimes want to know the original UNIX file- or directory name. The DOS Services `udir` command displays the contents of a UNIX directory in a format that combines the UNIX command `ls -l` and the DOS `DIR` command. The first two fields show both the UNIX name and its corresponding mapped DOS name.

The `udir -h` option displays "hidden" UNIX files. These are UNIX files with names that start with a period, which are normally not displayed in a directory listing. For example, this command displays the names of all files in the current directory, including hidden files:

```
C: udir -h
```

Using DOS drives

Drive letters are used under DOS Services the same way they are under raw DOS. However, DOS Services imposes certain additional conventions and limitations. Certain drive names should always be associated with certain devices, for instance.

Drive letters A: and B: should be used only with devices that are functionally equivalent to floppy disks. This includes physical floppies and virtual floppies.

Drive letters C: and beyond are used for devices which DOS Services treats as hard disk drives. These can include:

- The UNIX partition used to hold the hard disk filesystem shared by the DOS and UNIX systems.
- Virtual DOS partitions, which are portions of the shared filesystem set aside to emulate DOS disks.
- Physical DOS partitions on the same disk where the shared filesystem resides.
- Actual DOS-formatted hard disks which are separate from the hard disk used to hold the filesystem shared by the DOS and UNIX systems.

Drives C:, D: and J: are used to access the partition shared by DOS and the UNIX system.

Drive E: is the default designation for a physical DOS partition on the main disk (the same disk that holds the partition shared by the DOS and UNIX systems). The DOS partition is a special section of the fixed disk reserved for DOS files. If no physical DOS partition is present, E: can refer to a virtual DOS partition. See "Using physical DOS partitions" (page 156) for more about physical DOS partitions.

Drive letters E: through I: can be used to refer to either physical or virtual DOS partitions. No letters beyond I: should be used for virtual partitions.

DOS Services automatically allocates room for drives up to and including N:. If you need more, you must specify that with the **LASTDRIVE** command in the *CONFIG.SYS* files used in DOS image construction. (See *Administering DOS Services* in the *System Administrator's Guide* for details.)

Drive letters K: through LASTDRIVE can be used for other DOS devices, like CD ROM drives. The drivers for these devices are loaded in the *CONFIG.SYS* file. Starting with K:, DOS Services automatically assigns the next available drive letter to each such device.

15. Finding your way around with DOS

Another purpose for high drive letters is for use with the **SUBST** command. **SUBST** is often used to equate a long pathname string to a two-letter drive string.

```
$ subst m: d:\clients\reports\monthly\june
```

Having a series of assignments like the one above can save you a lot of key-strokes when doing operations that require hopping back and forth between directories.

For most purposes, DOS drives C: and D: are the most convenient drives to use when you install and run DOS commands and applications.

The following table illustrates these drive letter conventions under DOS Services and the sections which follow provide details.

Drive Letter	Used for
A:	Floppy disk (real or virtual)
B:	Floppy disk (real or virtual)
C:	Accesses the shared filesystem, starting at root.
D:	Accesses the shared filesystem, starting at user's <i>HOME</i> directory.
E:	Default designation for the physical DOS partition on the DOS Services fixed disk. Can also be used for virtual DOS partition.
F:	Available for virtual or physical DOS partition.
G:	Available for virtual or physical DOS partition.
H:	Available for virtual or physical DOS partition.
I:	Available for virtual or physical DOS partition.
J:	Accesses the shared UNIX/DOS filesystem starting at <i>\usr\ldbin</i> .
K: to LASTDRIVE	Available for other DOS devices.

Drive D:

Your own files and directories on the DOS Services fixed disk are accessible on drive D: just as they are on drive C:. On drive D:, however, your UNIX *\$HOME* directory is the *root* of the DOS filesystem. That is, if you are logged in as the user *ELAINE*, the directory *D:* contains the same files as *C:\USR\ELAINE*. Because your home directory is the root of the filesystem on drive D:, you cannot move upward. This means you can only use drive D: to access files in your home directory or the subdirectories beneath it.

Drive D: is useful for installing and running some DOS applications that modify or create files in the root directory. When you install such applications on drive D:, they modify or create files in your home directory rather than altering the systemwide root directory (C:\). See *Administering DOS Services* for further information on installing DOS applications.

Drive D: is unique for each user.

Drive E:

Drive E: gives you access to the *physical DOS partition*, if available. This is a special section of the fixed disk that is reserved exclusively for DOS work. Drive E: is usable only if the system administrator has created and formatted the DOS partition. UNIX files cannot be created on drive E: like they can on drives C:, D:, and J:. UNIX does not have direct access to DOS files created on drive E:. Although drive E: does not share the same files as drives C:, D:, and J:, you use it like a standard DOS disk drive.

Drive E: contains no DOS files when you first install DOS Services, but as you use your system, you can add DOS programs, files, and directories to drive E:.

Drive E: is the same for all users. By default, drive E: is a public resource. DOS files and directories created on drive E: are not owned by specific users or protected by UNIX file protection mechanisms. This means that all users can create files on drive E:, and all users have the power to remove or change any file.

Write access to drive E: is available on a first-come, first-served basis. As long as nobody is writing a file on drive E:, all DOS Services users can read any file on drive E:.

If your computer has multiple DOS partitions on several fixed disks, DOS Services checks them all. Drive E: always accesses the *first* primary partition DOS finds, which is usually on the first disk. For further information on the physical DOS partition, see *Administering DOS Services*.

Drive J:

The directory *J:* contains the same files as *C:\USR\LDBIN*. Because *\USR\LDBIN* is the root of the filesystem on drive J:, you can use the J: drive to install public applications that must be installed in the root directory.

Virtual DOS floppies and virtual DOS partitions

Your DOS Services system may have one or more virtual floppy drives or virtual DOS partitions. These are files within the shared filesystem that are formatted as DOS volumes. A virtual floppy is a UNIX file which emulates the function of a DOS floppy disk. You can store files there and even boot from it. A virtual DOS partition is a UNIX file which emulates the function of a DOS partition on the hard disk.

These virtual drives are not useful in the UNIX environment, but you can use them with DOS as you would use physical DOS diskette drives or physical DOS partitions. *Administering DOS Services* in the *System Administrator's Guide* describes how you create and administer virtual floppies and partitions.

By default, virtual floppies and partitions are not automatically accessible when you enter the DOS environment. You must use the `dos +a` option to attach any virtual floppies or partitions you want to use during a particular DOS session. See "Attaching devices" (page 152) for instructions.

Virtual floppies and partitions have the same access restrictions as the physical DOS partition (drive E:). Multiple DOS processes can read the same virtual floppy or partition at the same time, but when a process writes to the virtual device, no other process can read or write to the device until the writing process exits.

You can use the `dosopt` command as described in *Administering DOS Services* in the *System Administrator's Guide* to configure DOS applications or the DOS environment to attach specific virtual floppies or partitions automatically.

Reassigning DOS Services drives

Unless you intend to change standard DOS Services functionality, do not use the DOS `ASSIGN`, `JOIN`, or `SUBST` commands to redefine drives C:, D:, E: or J: so they refer to other drives or directories. You can, however, use these commands to make other DOS drives refer to the standard DOS Services drives without affecting DOS Services functionality. For example, the following command defines DOS drive M: so that it refers to the subdirectory `REPORTS\MONTHLY` within your home directory (`D:\`):

```
$ subst m: d:\reports\monthly
```


Chapter 16

Working with DOS files

DOS and the UNIX systems use different file naming conventions. See "Naming DOS files and directories" (page 132) for an explanation of the differences, along with instructions for working with both types of filenames at the same time.

DOS and the UNIX systems also store text files in different formats. The UNIX system stores text lines as a sequence of characters terminated by a line-feed character. DOS, on the other hand, terminates text lines with a carriage-return character followed by a line-feed character. A file created in one format can appear corrupted when accessed by the other.

Converting DOS and UNIX files

When you use DOS in Open Desktop, you can use any file that was created with DOS because these files are stored in DOS format even when they are created on the shared UNIX/DOS filesystem. To use a text file in UNIX format with DOS programs, however, you must convert the file to DOS text format using the DOS Services `unix2dos` command. For example, to convert the file `letter` in UNIX format to the file `ltr.dos` in DOS format, type:

```
C: unix2dos letter ltr.dos
```

You can also convert the file and copy it from one drive to another in one step, as the following example illustrates.

```
C: unix2dos c:bdgtmemo a:budget
```

When you create text files with DOS that you want to use later with UNIX utilities, you can convert them to UNIX text format with the DOS Services `dos2unix` command. For example:

```
C: dos2unix memo memo.unx
```

You can use `unix2dos` and `dos2unix` both in the DOS environment and from the UNIX shell. When you enter the `unix2dos` or `dos2unix` command in the DOS environment, you use DOS filenames, including mapped names when appropriate. When you use these commands from the UNIX shell, use UNIX (unmapped) names. The following example converts the file `message.tobob` (which would have a mapped name in the DOS environment) from UNIX format to DOS format and names the DOS file it creates with a legal DOS name:

```
$ unix2dos message.tobob message.bob
```

You can combine these commands with other DOS or UNIX commands through pipes and redirection. For example, the following command converts the file `names` from DOS format to UNIX format, sorts the text, and appends the sorted text to the UNIX file `newnames`:

```
$ dos2unix names | sort >> newnames
```

Do not specify the same name for the source file and the target file or try to redirect your output back into the source file. The following examples are incorrect:

```
$ dos2unix names names # incorrect
$ dos2unix names > names # incorrect
```

When you omit the target filename, `unix2dos` and `dos2unix` display the text file conversions but do not save them.

When you do not know the format of a text file, you can use the `unix2dos` or `dos2unix` command to convert to the format you need, just to be sure. The commands do not change anything when the file is already in the target format.

NOTE Use `unix2dos` and `dos2unix` only on ASCII text files. These commands do not convert programs, database files, or special-format files created by some word processors.

Accessing other users' files

DOS Services users have limited access to files owned by other users. Whether or not you can inspect or modify other users' files depends on how UNIX permission modes are set on your computer. See "Changing access permissions" (page 84) for UNIX access control. All DOS and UNIX files and directories you create or access in the shared UNIX/DOS filesystem are protected by these permission assignments.

DOS Services, unlike a conventional DOS system, is designed to accommodate multiple users. It therefore provides tools for preventing inspection, alteration, or execution of files by unauthorized users. In general, you cannot modify or delete a file or directory that belongs to someone else.

With DOS Services, you can restrict access to your files so that unauthorized users cannot see the contents of your directories or read your files. On the other hand, you can also grant other users permission to modify or delete your files and directories if you so choose. See "Changing access permissions" (page 84) and "Controlling access to files" (page 32).

The following default permissions are typical:

- Users can inspect the contents of any directory with the **DIR** command.
- Users can read the contents of any file (with the **TYPE** command, for example). Users can also copy any file to their own directories.
- Users can run programs contained in any directory.
- Users cannot modify or delete files or directories belonging to other users.

DOS applications and file permissions

Remember that most DOS applications are designed for a single-user environment. When used with DOS Services in a multiuser environment, most DOS applications do not protect your files from being simultaneously updated by you and another user with write permission.

You should consider carefully which combination of file and directory permissions give you the most appropriate protection. For example, to prevent a file from being simultaneously updated by someone else while you are working on it, you could temporarily remove execute permission for the directory containing that file for all other users. This would prevent anyone from even looking at the file until you were done. Alternatively, you could remove everyone else's write permission for the file. This would allow others to look at a file you are working on, but not to update it. Note that these measures do not protect a file if another user has opened the file and is using it at the time you change permission modes.

Newer DOS programs that use locking calls can prevent these problems without any special user action.

Printing from the DOS window

All standard DOS print functions work in DOS Services. These functions include the print screen ((**Prt Sc**)) key, the **PRINT** command, the **COPY** command, and printing operations performed by DOS applications.

By default, DOS Services sends DOS printer output via the UNIX print spooler to a printer named **doslp**. (Your system administrator must set this printer up before it can be used.) The following describes printing procedures you can use with the default DOS Services configuration.

Printing from DOS applications

DOS Services stores printing sent by DOS applications to any of the DOS parallel ports (LPT1, LPT2 or LPT3) in a temporary file. It is printed when either of two conditions occurs:

- You exit the application and return to your DOS prompt, or
- More than 15 seconds have elapsed since the application sent a character to be printed.

Printing with the DOS COPY command

You can print by using the DOS COPY command exactly as you would under standard DOS:

- `copy filename prn`
- `copy filename lpt1`
- `copy filename lpt2`
- `copy filename lpt3`

Printing with the DOS PRINT command

To print a file using the DOS PRINT command, type the command in the form:

- `print filename`

You cannot use PRINT options (such as */T*, */C*, and */P*) when you use the UNIX spooler.

Printing with the Prt Sc key

Press the *<Shift>* and print screen (*<Prt Sc>*) keys at the same time to print the current screen contents just as you would under standard DOS.

To use the *<Prt Sc>* key to save and print your screen contents as the screen is updated:

1. Press *<Ctrl><Shift><Prt Sc>* once to start saving your screen contents. You can then continue to perform operations that change the appearance of your screen. DOS Services saves all changes in a temporary file until you are ready to print.
2. Press *<Ctrl><Shift><Prt Sc>* a second time to stop the accumulation of screen contents and start printing.

The "printer not ready" message

Printing from some DOS applications may fail and produce an error message similar to the following:

```
Printer not ready.
```

To print from these applications, attach the printer directly to the DOS process:

1. If the printer is currently used to print spooled UNIX print jobs, the system administrator must use the **disable** command to disable the printer.
2. Use the **+a** option to attach the printer you want to use; see "Attaching devices" (page 152). DOS Services uses the device names *lp0*, *lp1*, and *lp2* to identify the first, second, and third parallel printer ports. Use the name that corresponds to the port to which your printer is attached. For example, if your printer is attached to */dev/lp0*, you can start the DOS environment with the command:

```
dos +alp0
```

Consult the manuals for your computer and the DOS Services */etc/dosdev* file if you are uncertain how to identify your printer port.

3. You can now start the application and send data to the printer.
4. When you are finished using the printer that is directly attached to DOS, the system administrator can reenable it for UNIX printing by using the **enable** command. UNIX printing cannot be enabled if the DOS process using the directly attached printer is still running.

Chapter 17

Controlling the DOS work environment

When you use Open Desktop DOS Services to run DOS commands and applications, you see the same behavior you would see on a conventional personal computer running the same commands and applications. This is possible because DOS Services creates a *virtual personal computer* (also called a *virtual PC* or *virtual machine*) for you whenever you run DOS. A virtual PC has all the important characteristics of a real stand-alone, single-user computer based on the Intel® 8086 processor. For example, DOS Services by default allocates 640K bytes of memory to your virtual PC. Any software running under the control of the virtual PC can use this memory the same way the same software would use 640K of memory on a stand-alone 8086 computer.

Open Desktop can create more than one virtual PC at a time, which allows users to run several DOS tasks at once. Each DOS environment under DOS Services runs in its own separate, protected, virtual machine, which cannot harm the operation of other DOS environments or the UNIX system. In particular:

- DOS programs cannot disable system interrupts. They can only disable their own "virtual" interrupts, which affect only that one DOS environment and not any other DOS environment or the UNIX environment.
- Errant DOS processes cannot damage UNIX processes or other DOS processes because each DOS environment is assigned a specific segment of memory and cannot write outside it.
- DOS programs can only affect I/O devices that are assigned to them, and not those assigned to the UNIX system or other DOS programs.

You can customize a virtual PC in much the same way you can customize a conventional stand-alone personal computer. For example, if you run DOS applications that need more than the default 640K bytes of memory, you can add expanded memory. If you use a DOS application that needs a COM port, you can add one. When you use DOS Services, however, you do not open

your computer and install adapter cards containing memory chips or a COM port. Instead, you use simple command options that tell DOS Services to configure these resources—which are already physically present—so they become part of your virtual personal computer. Because each virtual PC is independent of all others, you can customize each one as appropriate for the applications running in it.

DOS Services uses *DOS images* to improve efficiency. A DOS image is a frozen picture, or snapshot, of DOS after it has been loaded into memory and is running. This image includes information DOS needs about the virtual PC configuration. When you start DOS from your UNIX shell or from the Desktop, a virtual PC is created and a DOS image is loaded into that virtual PC's memory. This procedure has the same effect as booting DOS on a conventional personal computer, but it is much quicker.

Controlling the DOS window

You can control the the DOS window with the DOS menu.

There are two ways to invoke the DOS menu. To invoke the DOS menu with the mouse, move the cursor into the DOS window and press and hold the right mouse button. The DOS menu appears. Keep holding the button down and move the mouse cursor to the desired option on the menu, and then release the button. The option is selected and the DOS menu goes away.

You can also invoke the DOS menu by pressing a special key sequence. By default it is (Alt)D, but it can be redefined. The key sequence works only when the DOS window is already selected. To select one of the options on the menu, click on it with either the left or right mouse button. To close the DOS menu without selecting any option, simply click the mouse outside of the DOS menu.

The options on the DOS menu are as follows:

Zoom

Zooming a window means causing it to expand so it fills your whole screen. When you want to run a DOS EGA/VGA graphics program, you must zoom the window it is in; you cannot run such a program in a normal window. You can also zoom any other window if you want it to take over the screen. To unzoom (that is, return a zoomed window to its default size), press the DOS menu key sequence ((Alt)D by default).

- Focus [Unfocus]** Selecting the **Focus** option allows you to use the mouse with the DOS application that you are running in the DOS window, if that application supports a mouse. Selecting the **Unfocus** option allows you to use the mouse in other windows. Only one of these options appears on the **DOS** menu at a time. When the mouse is already focused in the DOS window, it cannot be used to bring up the **DOS** menu. You must use the **DOS** menu key sequence.
- Refresh** Selecting the **Refresh** option redraws the DOS window.
- DOS Colors [X Colors]** Selecting the **DOS Colors** option sets the colors for your DOS window to the sixteen standard text colors. Select the **X Colors** option sets it back to the colors chosen through the **Color** utility. Only one of these options appears on the menu at a time.
- Quit** Selecting the **Quit** option closes your DOS session.

Some applications run in graphics mode only part of the time. When using such an application, you only need to zoom it when it enters graphics mode. For example, a spreadsheet can work as a text application but it can also draw graphs. In a case like this, you could perform text entry in a regular DOS window without worrying about zooming. If your application enters graphics mode, however, a message displays reminding you to zoom, and the DOS window running your program becomes unusable until you have done so. You can return to the normal DOS window after you are finished using graphics mode by pressing the DOS menu key sequence (usually **(Alt)D**).

Note that you can unzoom at any time, even if you are still in EGA/VGA graphics mode, in order to use other clients. However, when you unzoom while your application is in EGA/VGA graphics mode, the application in the window is suspended until you zoom it again.

When the DOS program you are running requires a mouse, use the **DOS** menu key sequence; then select the **Focus** option from the **DOS** menu to focus your mouse in the DOS window. This means that your mouse input goes to the DOS program instead of to the server. The **Unfocus** option reverses this, so that your mouse input goes to the server. If you select the **Zoom** option, your mouse is automatically focused for you while you are zoomed.

Changing colors

By default, DOS Services uses the current UNIX color palette to display the colors in the DOS window. For many DOS programs this is sufficient. For DOS programs that depend heavily on color, the results can be peculiar. When the colors DOS Services expects to use are not available, you may find that objects in their DOS windows are displayed in unexpected colors.

The colors you see in your DOS program depend upon a number of factors: your hardware setup, the colors chosen for your X Windows session, and the color requirements of the DOS program you are running. DOS programs run in X windows may produce distorted colors or unreadable screens when run on 16-color servers.

The DOS Colors option on the DOS menu allows you to have your DOS windows displayed in true DOS colors. This can have an unexpected impact on the appearance of your UNIX windows. Alternatively, you could use the Desktop Color control (page 60) to select a special DOS palette. Some DOS applications let you select colors specifically for that application.

See *Administering DOS Services* in the *System Administrator's Guide* for more about controlling colors in the DOS window.

Using AUTOEXEC.BAT and CONFIG.SYS files

DOS interprets the commands in two special files automatically every time you enter the DOS environment. These files are *AUTOEXEC.BAT* and *CONFIG.SYS*.

You can use *AUTOEXEC.BAT* to customize your DOS environment or to run commands you want executed every time you use DOS. For example, if you run a program called *GRAPHS* every time you use DOS, you could include the command to run the program in your *AUTOEXEC.BAT* file.

The *CONFIG.SYS* file (if it exists) contains information about your computer's configuration that the system needs to know every time you run DOS. Some DOS applications, for example, require device drivers that are identified in *CONFIG.SYS*.

Cc

Because different users may want to include different commands in their *AUTOEXEC.BAT* or *CONFIG.SYS* files, DOS Services provides for both:

- System default *AUTOEXEC.BAT* and *CONFIG.SYS* files, which affect all users unless they explicitly specify otherwise.
- Personal *AUTOEXEC.BAT* and *CONFIG.SYS* files, which individual users can create to customize their own personal DOS environments.

If you create a personal *AUTOEXEC.BAT* file in your home directory, DOS Services executes it whenever you enter the DOS environment or start a DOS process. DOS Services executes your home directory *AUTOEXEC.BAT* file after executing the root directory *AUTOEXEC.BAT*.

In general, DOS Services interprets *CONFIG.SYS* commands just as conventional DOS personal computer does. However, the *FCBS* command is effective only when you use an actual DOS filesystem. It is not used on any portion of the shared UNIX/DOS filesystem. See *Administering DOS Services* in *System Administrator's Guide* for more about *FCBS*.

DOS Services does *not* interpret *BUFFERS* commands in any of your system's *CONFIG.SYS* files at DOS run time. The *BUFFERS* value is defined in the DOS images at the time they are created and cannot be changed unless you make new DOS images. The *BUFFERS* value used in the default DOS images is 15, the standard DOS default value for 640K of RAM. See *Administering DOS Services* in *System Administrator's Guide* for further information on changing *BUFFERS* and making new DOS images. The *BUFFERS* command is effective only when you use an actual DOS filesystem. It is not used when you access the shared DOS/UNIX filesystem.

Configuring memory

DOS 5.0 provides tools for maximizing the amount of Conventional DOS Memory available for your programs. These tools cannot be used on an 8086 computer, but you can use them under DOS Services. You can load device drivers and TSRs (Terminate and Stay Resident programs) into the Upper Memory Blocks (UMBs) and you can load DOS itself into the High Memory Area (HMA).

Most of this has already been done for you. By default, when DOS Services comes up it loads a special extended memory manager, *MERGEXMS.SYS*, which provides access to the Upper Memory Blocks. It also loads DOS into the High Memory Area.

MERGEXMS.SYS is the only Extended Memory Manager that can be used with DOS Services. Do not use *HIMEM.SYS*, *XMS.SYS*, or any other DOS extended memory manager produced by third-party vendors.

As in standard DOS, once the Upper Memory Blocks have been enabled, you can free up DOS Conventional Memory by loading your device drivers and TSRs there. Refer to your DOS manuals for more about DOS memory management.

To load device drivers and TSRs in the Upper Memory Blocks, use commands such as the following in any *CONFIG.SYS* file which will be interpreted when DOS Services starts:

```
DEVICEHIGH=DEVICE1.SYS  
DEVICEHIGH=DEVICE2.SYS  
INSTALL=TSR1.EXE  
INSTALL=TSR2.EXE
```

To load TSRs into the Upper Memory Blocks, you can also use commands such as the following in any *AUTOEXEC.BAT* file which will be interpreted when DOS Services starts:

```
LOADHIGH TSR1.EXE  
LOADHIGH TSR2.EXE
```

Using expanded memory (EMS)

DOS Services supports the Lotus/Intel/Microsoft Expanded Memory Specification (EMS), so you can run any DOS applications that use expanded memory and conform to this specification.¹ DOS Services expanded memory is available in the following sizes: 512 Kbytes, and 1, 2, 3, 4, 5, 6, and 8 Mbytes. The default amount of expanded memory is one megabyte, but you can easily request any of the allowable values. (You should not request more memory than you need, however, because it wastes system resources.)

Your computer does not need to have actual physical memory in the amount you request when you use expanded memory, and you do not need an EMS memory card to use expanded memory with DOS Services. DOS Services simulates expanded memory by using standard UNIX system virtual memory. Provided you have at least the minimum amount of memory required to run DOS Services, you can use any of the expanded memory values that DOS Services supports.

1. *The Lotus/Intel/Microsoft Expanded Memory Specification*, Version 4.0, Lotus Development Corporation, Intel Corporation, and Microsoft Corporation.

To use expanded memory, you must request it with the DOS **+a** option. To request the default amount of one megabyte of expanded memory, use the **+aems** option. For example:

```
dos +aems  
dos +aems 123
```

The first example starts a DOS environment with one megabyte of expanded memory. The second example starts Lotus 1-2-3 with one megabyte of expanded memory.

To request a different amount of expanded memory, use one of the following **+a** options:

DOS option	Memory
+aems512	512 Kbytes
+aems	1 megabyte
+aems1	1 megabyte
+aems2	2 megabytes
+aems3	3 megabytes
+aems4	4 megabytes
+aems5	5 megabytes
+aems6	6 megabytes
+aems8	8 megabytes

For example, to request four megabytes, type:

```
dos +aems4
```

Note that **+aems1** has the same effect as **+aems**.

Using peripheral hardware with DOS Services

Because DOS Services is a fully configurable environment, you may run many different kinds of DOS sessions. You may configure DOS Services to run in VGA mode with one program and in CGA mode for another. You might run one program with minimal memory to conserve resources and allocate 5 megabytes of EMS memory to another. You might attach a local dot-matrix printer when you use your database program, while you attach a network laser printer for desktop publishing.

When you use many different DOS programs you usually run many different DOS environments. Many users set things up so that the devices they need are automatically attached whenever a particular DOS program is run. Since this device attachment takes place automatically and invisibly, it is common to forget exactly which devices are available at the moment.

DOS

Using the device information window

The Device Information window gives you instant access to this information any time you are in DOS. It works from the DOS command line or from within any DOS application. It "pops up" when you hit a hot-key sequence, like a TSR program in the standard DOS environment.

To open the Device Information window, press (Ctrl)(Esc), then (Ctrl)I.

While the Device Information window is displayed, your DOS process is suspended. Any other processes, (DOS or UNIX) which may be running in the background continue to run. When you have obtained the information you need, press (Esc) or (Space) and you pop back into your DOS session, which picks up exactly where you left it.

The Device Information window is only available when your DOS window is in text mode.

The default color scheme is yellow characters on a red background. If you find this hard to read or prefer another arrangement, it is configurable. You can reset the colors of the Device Information window by setting the UNIX environment variable, DOSCONFIG. The following example shows how this is done from the Bourne shell.

```
# DOSCONFIG=menucolor.white.blue
# export DOSCONFIG
```

The `menucolor` option allows you to specify two parameters, separated by dots. The first parameter is the foreground color; the second is the background color. The code above gives you white characters on a blue field. Sixteen colors are available for each. See *Administering DOS Services in the System Administrator's Guide* for a list.

Attaching devices

If you want to use a hardware device that is not automatically available when you use DOS, you request access to it using the `dos +a` ("attach device") option in the form:

```
dos +adevice_name [command]
```

The command form `dos +a`*device_name* starts a DOS environment and attaches the requested device to the DOS process so you can use it for the duration of the DOS environment. The command form `dos +a`*device_name* *command* attaches the specified device to the DOS process and also runs the specified DOS command. You can then use the specified device for the duration of the program you start with *command*. If your specified device is not available (typically because another UNIX or DOS process is using it), DOS Services displays a message informing you that you cannot use the device.

To attach more than one device to a DOS process, use more than one **+a** option. For example:

```
dos +acom1 +aems
```

Examples illustrating the use of the **+a** option appear throughout this chapter. See *Administering DOS Services in System Administrator's Guide* for descriptions of other useful procedures, including the use of the **dosopt** command to configure DOS commands so they automatically request required devices.

Using display adapters and serial terminals

DOS Services automatically senses the type of display adapter you use in the system console and properly displays DOS processes. DOS Services is compatible with VGA, EGA, CGA, Hercules, and monochrome display adapters. When you use a serial terminal, DOS Services displays DOS processes as though they are running on a monochrome console. You can use the **+a** option to specify explicitly a particular display type, but this is normally unnecessary. For example:

```
dos +acga
```

(Other displays are designated with **+avga**, **+aherc**, and **+amono**.)

Using a mouse

Note that you should not modify any *CONFIG.SYS* files to identify a mouse driver as you would on a conventional personal computer running standard DOS. DOS Services uses a special mouse driver that is identified in the system default *\CONFIG.SYS* file.

DOS Services causes DOS to view any properly configured mouse as though it is a Microsoft Bus Mouse. If you install DOS applications that need to know about the specific mouse you use, always refer to it as a Microsoft Bus Mouse.

Using a modem

You can use either an external modem (one attached to a serial port) or an internal modem (one that requires an internally installed card) with DOS Services. If you have a choice, consider that external modems are easier to troubleshoot should problems arise.

For either kind of modem, install the modem by following the manufacturer's instructions to connect it to a serial port. Note that an internal modem generally replaces COM1 or COM2. To use the modem, attach the appropriate COM port to your DOS process by using the **+a** option when you start DOS. For example:

```
dos +acom1
```

For further information, see "Using COM ports" (page 154).

Using COM ports

DOS can use the COM1 and COM2 serial ports (equivalent to the UNIX devices */dev/tty1a* and */dev/tty2a*). The COM3 and COM4 ports are not supported. Only one DOS process at a time can use each COM port. To use a COM port, you must explicitly request access to it using the **+a** option when you start DOS.

DOS Services can attach COM ports in two different ways: indirectly or directly. You do not need to understand the technical distinction between these two forms. However, you must choose one form or the other when you start DOS. Consider these trade-offs as you make your choice:

- Indirect attachment is more reliable, but when the system is heavily loaded, it may be slower than direct attachment. Try this form of attachment first if you are uncertain which to use.
- Direct attachment is faster but less reliable than indirect attachment when the system is heavily loaded.

Indirect attachment

To attach a COM port indirectly, use the **+acom1** or **+acom2** option. For example:

```
dos +acom1
dos +acom2 xtalk
```

The first example starts a DOS environment and requests access to COM1. The second example starts the CROSSTALK[®] application and requests access to COM2. In both examples, if the requested COM port is not available, DOS Services does not start DOS and instead displays an error message.

Direct attachment

To directly attach COM1 or COM2, use the **+adcom1** or **+adcom2** option. For example, to start a DOS environment and directly attach COM1, type:

```
dos +adcom1
```

To start CROSSTALK and directly attach COM2, type:

```
dos +adcom2 xtalk
```

Using COM ports to transfer files

You can use both directly and indirectly attached COM ports to transfer files between computers. However, the reliability of the transfer depends on many factors including line quality, transfer speed, and system load. If you use COM ports to transfer files at speeds greater than 4800 baud, use an error-correcting protocol to perform the transfer. Error-correcting protocols help ensure the integrity of data during transfer.

Using the game port

To use the game port, use the **+agame** option. For example:

```
dos +agame
```

Only one DOS process at a time can use the game port.

For further information on installing and configuring hardware devices, refer to *Administering DOS Services* in the *System Administrator's Guide* or consult your system administrator or DOS Services distributor.

Using virtual partitions and virtual floppy disks

Virtual DOS partitions and virtual floppy disks are UNIX files that contain actual DOS filesystems. By default, these virtual devices do not exist.

Virtual partitions and floppy disks are typically not of interest to most DOS Services users. If your computer does not have a physical DOS partition, however, you may find a virtual partition to be useful. Refer to *Administering DOS Services* in the *System Administrator's Guide* for further information on creating these virtual devices and on their characteristics.

To use a virtual partition or floppy disk, attach it to your DOS process using the **+a** option in the form:

```
+adrive_letter:=unix_file_name
```

You should normally use drive letters **a** or **b** to access a virtual floppy. Use drive letters **e**, **f**, **g**, **h** or **i** to access virtual partitions.² The full pathname of the UNIX file that contains the virtual partition or floppy drive is *unix_file_name*. For example, the following command starts a DOS environment and attaches the virtual partition */usr/fred/vdisk* as DOS drive F:

```
dos +af:=/usr/fred/vdisk
```

You can then access any DOS files contained within */usr/fred/vdisk* via DOS drive F:. You can change your current drive to drive F: with the command:

```
f: f:
```

You can list the files on drive F: with the command:

```
f: dir f:
```

² You may want to avoid using drive E: since it is assigned to the primary DOS partition by default. However, you can use drive E: if you wish. Note that if you assign drive E: to a virtual floppy or virtual partition, you will no longer have an attachment to the primary DOS partition (unless you specifically assign another letter to it).

To start a DOS environment and attach a virtual floppy named `/usr/phyllis/vflop`, use a command such as:

```
$ dos +ab:=/usr/phyllis/vflop
```

When you issue this command, the virtual floppy is accessible as DOS drive B, just as if it were a physical diskette drive. Note that if there is a physical drive B, you will no longer have access to it.

Virtual partitions and floppy drives have the following limitations:

- While multiple users can simultaneously attach and read a virtual partition or floppy drive, only one user at a time can write to a virtual partition or floppy drive.
- When one user is writing to a virtual partition or floppy drive, all other users are prevented from reading and writing to that device until the DOS session on the device is terminated.

See *Administering DOS Services* in the *System Administrator's Guide* for further information on attaching virtual partitions and floppy disks, including instructions on attaching them "exclusive" so they cannot be written by other users.

Using physical DOS partitions

A physical DOS partition is a portion of the fixed disk formatted as a DOS file-system and reserved exclusively for DOS files. DOS Services may have just one physical DOS partition (called the primary DOS partition); it may have both a primary and an extended DOS partition; or it may have no physical DOS partitions.

If your Open Desktop computer has a primary DOS partition, it is automatically available as DOS drive E: whenever you run DOS. If your computer has an extended DOS partition, you must attach each logical drive you want to use to an available DOS Services drive letter. To attach a logical drive that is within the extended DOS partition, use the `+a` option in the form:

```
+aDOS_Services_drive_letter:=doslogical_drive_letter
```

`DOS_Services_drive_letter` can be e, f, g, h or i.³ `logical_drive_letter` can be any of the logical drives available under raw DOS.

3. You may want to avoid using drive E: since it is assigned to the primary DOS partition by default. However, you can use drive E: if you wish. Note that if you assign drive i: to a logical drive in the extended partition, you will no longer have an attachment to the primary DOS partition (unless you specifically assign another letter to it).

For example, the following command attaches your system's logical DOS drive D: to the DOS Services F: drive:

```
dos +af:=dosd
```

Refer to the file */etc/dosdev* for a list of available logical drives.

Multiple DOS processes can *read* files on a DOS partition at the same time, but only one process at a time can *write* to the primary DOS partition or to a logical drive within the extended DOS partition. As soon as one process attempts to write to a DOS partition, no other process can read or write to the partition until the DOS session writing the first process is terminated.

Refer to *Administering DOS Services* in the *System Administrator's Guide* for further information on using and administering physical DOS partitions.

Chapter 3. Accessing Your Network With Telnet

Introduction.....	3-3
Getting Started with TELNET	3-3
Using TELNET Commands: A Sample Session	3-5
Emulation Options.....	3-6
Changing the Escape Character	3-7
Starting a Temporary DOS Shell.....	3-7
Summary.....	3-8

accessing your network with telnet

Introduction

This chapter tells you how to use the TELNET terminal emulator. TELNET allows you to log in to a remote host from your personal computer and work as if you were on a directly connected terminal. TELNET offers features of the following terminal emulation types: ANSI-standard X3.64, Heath-19, and DEC VT52.

This chapter assumes that you have installed and configured Locus TCP/IP For DOS as described in Chapter 2, that you have an account on a remote host, and that you have a HOSTS file on your personal computer that lists your computer and the available hosts on your network.

Getting Started With TELNET

TELNET is simple to use. To start a TELNET session and contact a remote host, type the following at the DOS C> prompt:

```
telnet burmese
```

where **burmese** is the name of the remote host you want to log in to.

After you open a TELNET session, information similar to the following appears at the bottom of your screen:

```
ansi  Mon Aug 02 16:20  Escape char: '^]'  burmese
```

This is the TELNET status line. The left-most field indicates the type of emulation requested, in this case ANSI-standard X3.64, which is the default. The second field displays the date and time, the third field displays the escape character in quotes

accessing your network with telnet

(in this case CTRL-]), and the last field displays the name of the remote host, burmese.

This information stays at the bottom of your screen until you exit TELNET.

Once TELNET has established a connection to the remote host, burmese, you are prompted to log in to burmese.

If a password is required, burmese prompts you for it as well. After you log in to the remote host, you can work as though you were directly connected via a terminal. You can create files, remove files, edit files, and send and receive mail on the remote host.

There are two ways to exit TELNET:

- Log out at the remote host prompt and press ENTER, which exits TELNET and displays the following message:

```
telnet: - connection closed
```

before returning you to the DOS prompt.

or

- Press

```
CTRL-]
```

at the remote host prompt which returns you to TELNET command mode, indicated by the `Locus telnet>` prompt.

At the TELNET prompt, type `bye`. This returns you to DOS.

Using TELNET Commands: A Sample Session

The previous section explained the TELNET status line, showed you how to start a TELNET session, how to log on to a remote host, and how to exit a TELNET session.

This section shows you how to use some of the commands to TELNET. You can only use the TELNET commands at the **Locus telnet>** prompt. These commands let you specify a different type of terminal emulation, set the escape character to be something other than CTRL-], close your current connection to a remote host and open a new session, and exit TELNET.

When you make any of these modifications, the TELNET status line reflects those changes.

The following sample session illustrates how Dick, our sample user, uses TELNET. Dick has Locus TCP/IP For DOS loaded on his IBM PC-compatible personal computer, which is running PC-DOS, Version 3.3. Dick decides to connect to **siamese**, which is one of the UNIX hosts on his network.

At the DOS prompt, Dick types:

```
telnet
```

The status line appears at the bottom of the screen, and he sees the TELNET prompt. To connect to **siamese**, Dick types:

```
connect siamese
```

at the TELNET prompt. When **siamese** prompts him, Dick logs in.

accessing your network with telnet

Emulation Options

Dick's status line indicates that he is using ANSI-standard X3.64 emulation, and he wants to change it to Heath-19 emulation. Dick has to be at the TELNET prompt to do this. So, at the UNIX prompt, he presses CTRL-] to leave UNIX and return to the **Locus Telnet>** prompt.

At the TELNET prompt Dick types:

```
h19
```

and presses ENTER. TELNET returns him to the UNIX prompt and indicates the change on the status line as follows:

```
h19  Mon Aug 02 16:20  Escape char: ^]  burmese
```

Because Dick is using the UNIX system, he has to set the UNIX **TERM** environment variable to identify the terminal type to the host. To do this, at the UNIX Bourne shell prompt Dick types:

```
TERM=h19
export TERM
```

The options to change TELNET emulation are as follows:

ANSI	ANSI-standard X3.64 (the default)
VT52	DEC VT52 terminal emulation
H19	Heath-19 terminal emulation

Changing The Escape Character

The escape character, listed in the third field of the status line, lets you return to the TELNET prompt without closing your login session on the remote host.

Dick decides he doesn't like using the default CTRL-] escape sequence because he has to use both hands to do it, so he decides to change it to CTRL-R.

First he escapes to the TELNET prompt by pressing CTRL-]. When he gets the TELNET prompt, he types `escape`, then presses ENTER. Now he presses CTRL-R and then presses ENTER again. TELNET lets him know the change has been successful:

```
Escape character is '^R'
```

Dick's status line indicates the change as well. Dick is returned to the UNIX prompt.

Now when Dick wants to return to the TELNET prompt, he can press CTRL-R.

Starting A Temporary DOS Shell

TELNET allows users to temporarily suspend a TELNET session and return to the DOS prompt.

To do this, Dick types his new escape sequence, CTRL-R, at the UNIX prompt, which returns him to TELNET command mode. To get to the DOS shell without terminating the TELNET session, Dick types:

accessing your network with telnet

presses ENTER, and he returns to the DOS prompt. While he is in DOS, Dick can use any DOS commands or run any DOS programs he pleases.

When he wants to return to TELNET, he simply types:

```
exit
```

at the DOS prompt and presses the ENTER key.

Summary

This chapter presented the following TELNET commands:

CONNECT	Connects you to the remote host you specify.
BYE	Exits TELNET.
H19	Requests Heath-19 emulation.
ESCAPE	Changes the escape character.
!	Enters the DOS shell.

For a brief summary of all the TELNET commands, type `help` or `?` at the TELNET prompt.

For a more detailed summary of what TELNET can do, refer to the description of TELNET in Appendix A, Reference.

accessing your network with telnet

Chapter 4. Transferring Files With FTP

Introduction.....	4-3
Getting Started with FTP	4-3
Using FTP Commands.....	4-4
Transferring Files between a Personal Computer and a Remote Host	4-5
Transferring Multiple Files between a Personal Computer and a Remote Host	4-6
Changing Directories in FTP	4-8
Suspending an FTP Session and Returning to DOS..	4-10
Exiting FTP.....	4-10
Summary	4-11

transferring files with ftp

Introduction

This chapter tells you how to use the FTP file transfer program. With FTP, you can transfer single or multiple files to and from remote hosts, transfer both ASCII and binary files between your DOS computer and a remote host, and temporarily suspend your FTP session.

This chapter assumes that you have installed and configured Locus TCP/IP For DOS as described in Chapter 2, that you have an account on a remote host, and that you have a HOSTS file on your personal computer listing the available hosts on your network.

Getting Started With FTP

FTP is simple to use.

1. To start FTP and connect to a remote host, type the following at the DOS prompt:

```
ftp rex
```

where `rex` is the name of the host you want to contact.

Messages similar to the following appear on your screen:

```
Locus Computing Corporation PC FTP(c) 1988, 1989
Trying ...
220 rex FTP Server (Ver. 4 Tue Aug 1 10:17 EDT 1988) ready.
```

The remote host, `rex`, prompts you for your user login name as follows:

```
User:
```

transferring files with ftp

2. Type in your user login name and press ENTER.

The remote host prompts you for your password:

```
331 Password required for user_name.  
Password:
```

3. Type your password and press ENTER. When you see the following prompts, FTP is ready to accept commands:

```
230 User user_name logged in.  
Locus PC FTP>
```

Using FTP Commands

The previous section showed you how to open an FTP session and log in to a remote host.

This section tells you how to use some of the commands to FTP. You can only use the FTP commands at the FTP prompt, **Locus PC FTP>**. These commands let you connect to a host, change directories on your personal computer or the remote host, transfer files between your personal computer and a remote host, and return to DOS.

The following sample session illustrates how Suzanne, our sample FTP user, uses this program. Suzanne has Locus TCP/IP For DOS loaded on her IBM PC-compatible personal computer, which is running DOS.

Suppose Suzanne starts an FTP session as described above, but when she types `ftp rex` at her DOS prompt, the host she requests, `rex`, is not available:

```
Trying ...  
Connection to rex failed
```


transferring files with ftp

FTP returns her to the FTP prompt. She can now use the **CONNECT** command to try another host. For example:

```
connect burmese
```

tells FTP that she wants to try to connect to host **burmese**.

After Suzanne has made a successful connection, she is prompted to log in to the remote host.

Transferring Files Between A Personal Computer And A Remote Host

Now that she is logged in, Suzanne wants to transfer some text files between her personal computer and the remote host, **burmese**. FTP transfers all files in ASCII text format by default. Since Suzanne wants to transfer text files, the default mode works fine for her.

If Suzanne wants to transfer executable programs, image files, or other special nontext files, she would have to specify binary transfer mode. For more information on FTP's file transfer modes, refer to the description of FTP in Appendix A, Reference.

To transfer a file from her local directory to the remote host, Suzanne types:

```
put taxes89.txt
```

This copies TAXES89.TXT onto the remote host and gives it the same name as it has on Suzanne's local personal computer. To give it a different name, Suzanne types:

```
put taxes89.txt fed.taxes89
```

transferring files with ftp

where `fed.taxes89` is the new name for the file on the remote host. The following message prints on Suzanne's screen to let her know the file transfer succeeded:

```
200 Port command okay.  
150 Opening data connection for fed.taxes89 (xxx.x.xxx,xxxx).  
226 Transfer complete.  
5395 bytes transferred in 2 seconds (2689 bytes/s)
```

To copy a file from the remote host to her local drive, Suzanne uses the FTP GET command, which uses the same syntax as PUT. For example, to get a copy of a file called `receipts` from the remote host and copy it to the local drive, she types:

```
get receipts
```

The file `receipts` gets copied into Suzanne's current working directory on her personal computer.

When she wants to rename the copied file in its new location, she uses the same syntax as for the PUT command described above. For example, if the file she wants to get from the remote host has a name that does not conform to DOS naming conventions, she might want to rename it as she copies it, as follows:

```
get receipts.back receipts.bak
```

Transferring Multiple Files Between A Personal Computer And A Remote Host

Suppose Suzanne wants to copy all files on the remote host ending with `.txt` on the remote host to her personal computer, or vice versa. The MGET and MPUT commands allow her to do this.

transferring files with ftp

The syntax for MGET and MPUT is the same as for GET and PUT, except she cannot rename files during multiple file transfers.

For example to transfer all files on her personal computer ending with .TXT to the remote host, Suzanne types:

```
mput *.txt
```

The * is a wildcard character that represents all the other letters in the file names.

FTP then prompts:

```
a(uto) p(rompt) ?
```

When Suzanne types a for "auto," all files ending with .TXT get copied to the remote host. When Suzanne types p for "prompt," FTP asks her to confirm each file transfer as follows:

```
put receipts.txt (y/n)?
```

If the multiple file transfer is successful, messages similar to the following appear:

```
200 Port command okay.
150 Opening data connection for boat.txt (xxx.x.xxx,xxxx).
226 Transfer complete
9529 bytes sent in 4 seconds (2382 bytes/s)
200 Port command okay.
150 Opening data connection for curtain.txt (xxx.x.xxx,xxxx).
226 Transfer complete
6434 bytes sent in 3 seconds (2144 bytes/s)
```

For more information on these commands, refer to Appendix A, Reference.

transferring files with ftp

Changing Directories In FTP

As soon as Suzanne has logged on to the remote host, she can change directories and see directory listings on both the remote host *and* her local directory.

On The Remote Side

When Suzanne wants to change directories on the remote host, she uses the CD command. Suppose Suzanne wants to change from her home directory to a subdirectory called *purchases/house*; she types the following at the FTP prompt:

```
cd purchases/house
```

When the command is successful, FTP prints the following message on Suzanne's screen:

```
200 CWD command okay.  
Locus PC FTP>
```

To view the contents of *purchases/house* with FTP, Suzanne types:

```
dir
```

and FTP prints the contents of *purchases/house* on Suzanne's screen:

```
200 Port command okay.  
150 Opening data connection for /lbin.ls (9.200.89,219) (0 bytes).  
total 24  
-rw-r--r-- 1 bin      1625    Feb 3   13:43 fabric.chair  
-r--r--r-- 1 bin     18353    Jul 3   10:12 cd.player
```

transferring files with ftp

On The Local Side

While she is logged into the remote host, Suzanne can still change directories and see directory listings on her personal computer.

For example, suppose Suzanne wants to copy a file called RADIO.TXT on her local drive to the remote host, but RADIO.TXT is located in a subdirectory called ELECTRIC\TOYS.

To get to ELECTRIC\TOYS, while connected to the remote host, Suzanne uses the LCD (local change directory) command as follows:

```
lcd electric\toys
```

FTP lets her know if the directory change has been successful:

```
FTP: Current working dir is C:\ELECTRIC\TOYS  
Locus PC FTP>
```

To view the contents of C:\ELECTRIC\TOYS, she types:

```
ldir
```

To transfer RADIO.TXT to the remote host, she types:

```
put radio.txt
```

transferring files with ftp

Suspending An FTP Session And Returning To DOS

Suzanne wants to view the contents of RADIO.TXT on her local drive, but she doesn't want to log out of her FTP session yet.

To do this, she types the following at the FTP prompt:

and she is returned to the DOS prompt. Now she can use the DOS TYPE command to see the contents of the file RADIO.TXT:

```
type radio.txt
```

To return to FTP, at the DOS prompt, she types:

```
exit
```

and presses ENTER.

Exiting FTP

To close an FTP session, Suzanne types:

```
bye
```

at the FTP prompt. This closes the connection to the remote host and returns her to the DOS prompt.

Summary

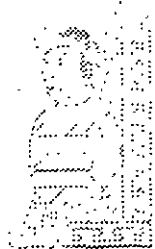
This chapter presented the following FTP commands:

!	Temporarily suspends the FTP session and returns you to the DOS shell.
?	Prints out brief description of FTP commands.
BYE	Closes FTP session and returns to DOS.
CD	Changes the remote directory.
DIR	Lists the contents of the remote host directory.
LCD	Changes the personal computer working directory.
LDIR	Lists the contents of the personal computer directory.
CONNECT	Requests a connection to a remote host.
GET	Gets a files from the remote host and copies it to your personal computer.
PUT	Copies a file from your personal computer to the remote host.
MGET	Gets multiple files from the remote host and copies them to your personal computer.
MPUT	Copies multiple files from your personal computer to the remote host.

For a brief summary of all of the FTP commands do,
transferring files with the
FTP prompt.
For a more detailed description of what FTP commands do,
refer to the description of `ftp` in `man ftp`, Reference.

INSTALACIÓN Y MANEJO DE REDES (LAN) EN PLATAFORMA UNIX

8.- CONECTIVIDAD



Mayo de 1996.

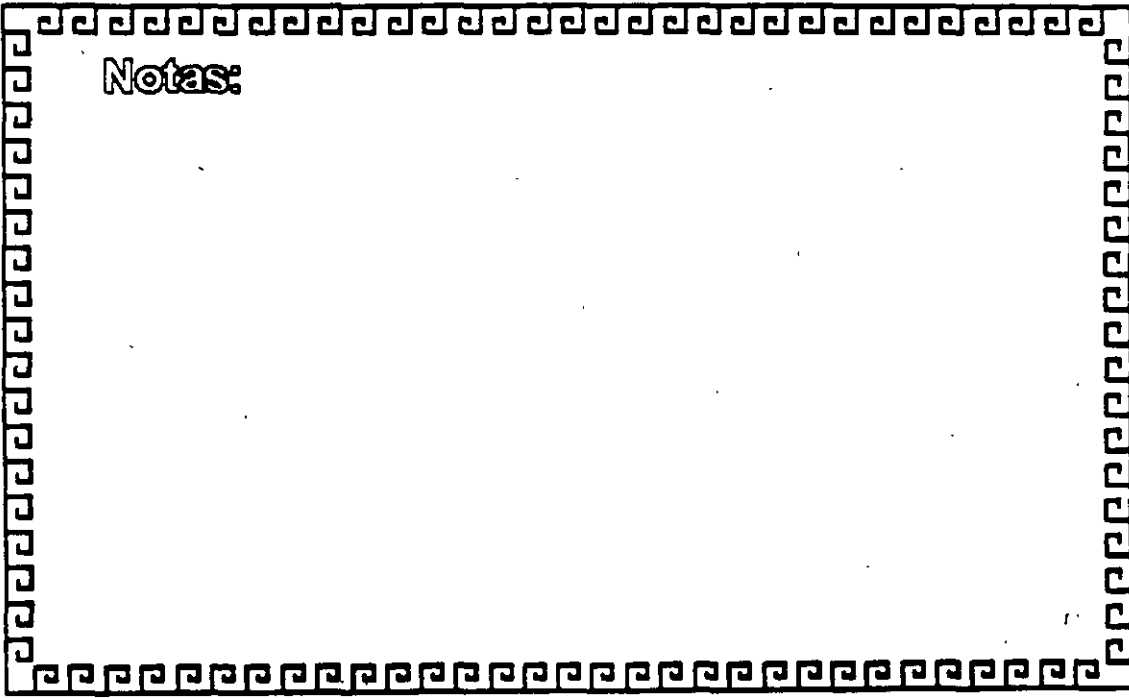


8.- CONECTIVIDAD

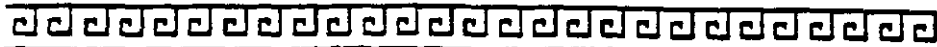


INTRODUCCION

TCP/IP Y LA INTEGRACION
DE AMBIENTES
HETEROGENEOS



Notas:

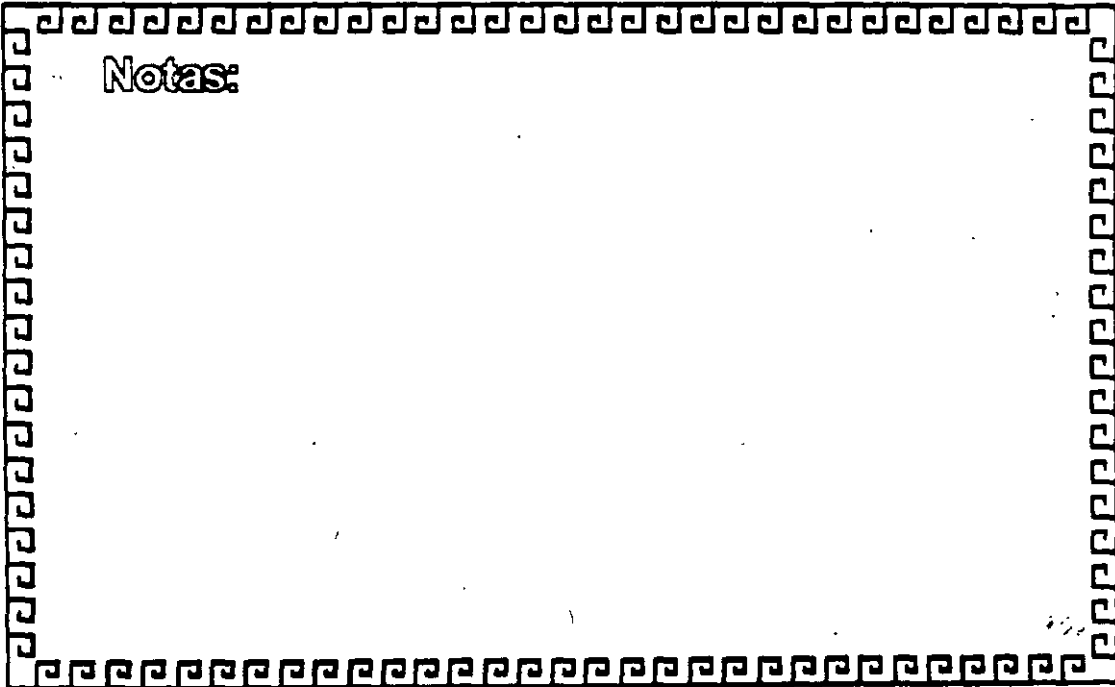


AGENDA

FAMILIA TCP/IP

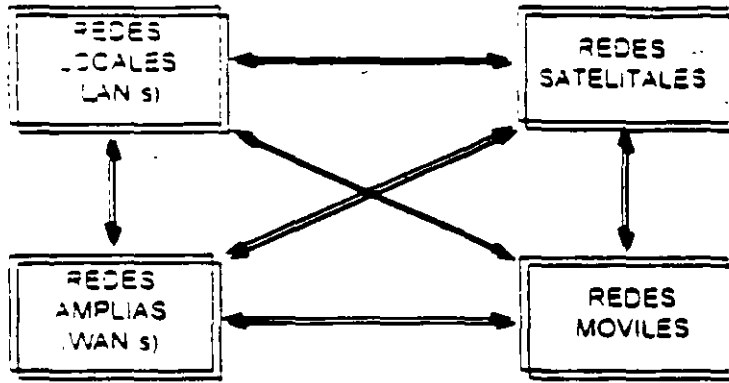
**PRODUCTOS Y ESTRATEGIAS
DE INTEGRACION**

CASOS DE ESTUDIO



Notas:

HISTORIA Y GENERALIDADES



Notas:



HISTORIA Y GENERALIDADES

- 1969 Empezar el trabajo con ARPANET
- 1972 Primera demostración de ARPANET
- 1976 Empezar la implantación de TCP/IP
- 1980 Se libera TCP/IP con Unix 4.1 BSD (BERKLEY)
- 1982 TCP/IP reemplaza a NCP en ARPANET
- 1983 Se publica TCP/IP con especificaciones Militares Estándares
- 1984 Se separa Milnet de Arpanet
- 1989-90 Mas de 200 proveedores soportan TCP/IP mas de 600 000 sistemas



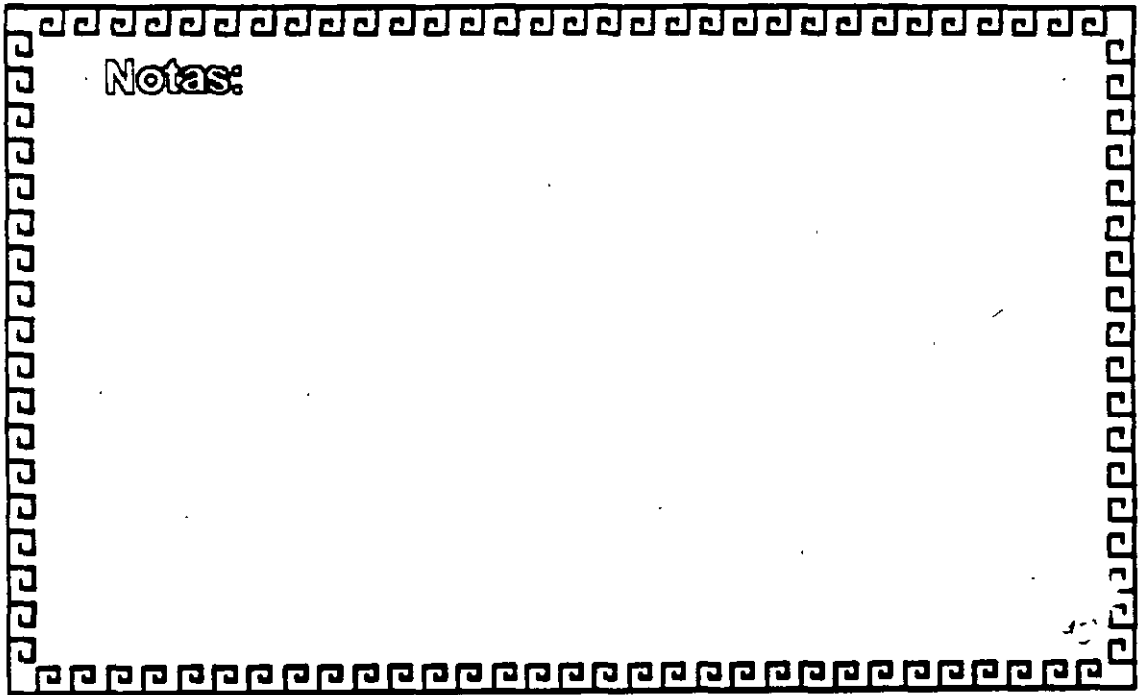
Notas:



HISTORIA Y GENERALIDADES

¿POR QUE TCP/IP?

- * Aceptado ampliamente por los centros de investigación y desarrollo en todo el mundo
- * Desde 1984 fue requerido por el gobierno y la defensa de los E U A
- * Los sistemas basados en Berkeley-Unix lo proveen
- * SUN (Sun Microsystems) le da a TCP/IP un posicionamiento comercial
- * Los ambientes mas técnicos adoptan TCP/IP.
- * Son los únicos protocolos realmente abiertos y estandares disponibles actualmente
- * Predecesores de los protocolos ISO



Notas:



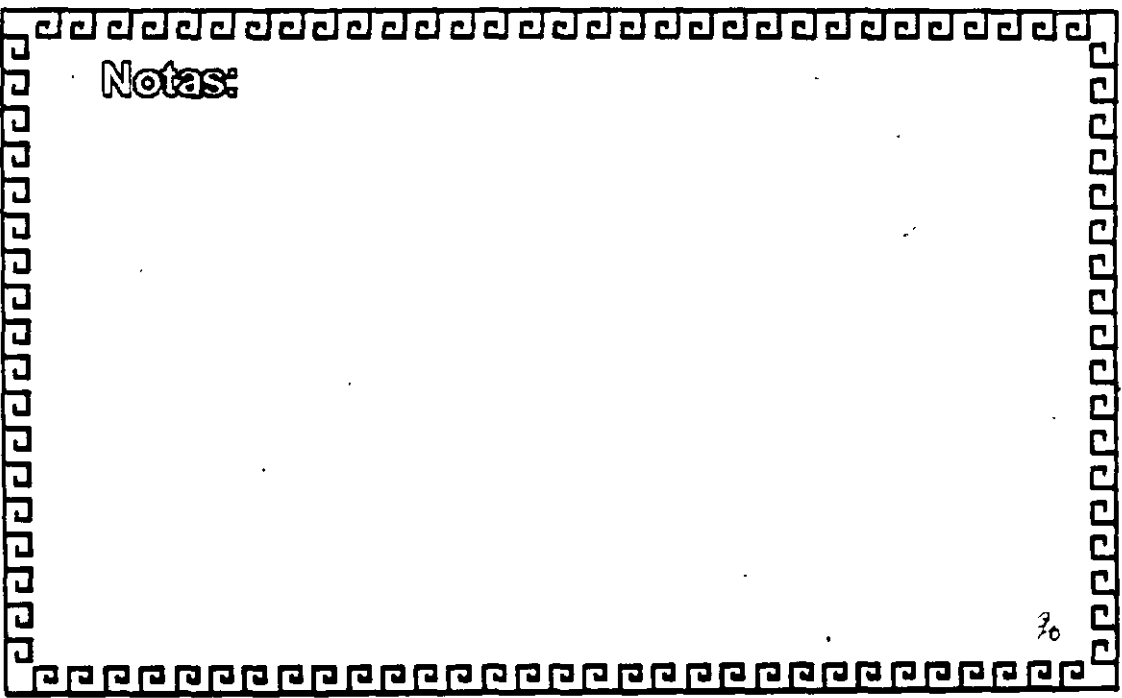
ARQUITECTURA

MODELO OSI

CAPA	NOMBRE
-	APLICACION
5	PRESENTACION
5	SESION
4	TRANSPORTE
3	RED
2	ENLACE
1	FISICO



Notas:



30



TCP/IP

ARQUITECTURA

NOMBRE	ARPANET	ARPANET REVISADO
Aplicación	Proceso a Proceso	Servicios de Aplicaciones
Sesión	Host - Host	
Transporte	Inter Red	Transporte
Red	Acceso a la Red	Red
Enlace		Enlace
Fisico		Fisico



Notas:



TCP/IP

ARQUITECTURA

TCP y OSI		
7 Aplicación	Servicios de Aplicación	SMTP
6 Presentación	Formatos de Datos	FTP
5 Sesión	Regulación de Conversación	Telnet, etc.
4 Transporte	Integridad de datos de extremo a extremo	TCP, UDP, etc.
3 Red	Ruteo y conmutación	IP, ICMP, etc.
2 Enlace	Transmisión de Frames	Ethernet,
1 Físico	Acceso al medio	Arpanet, etc.



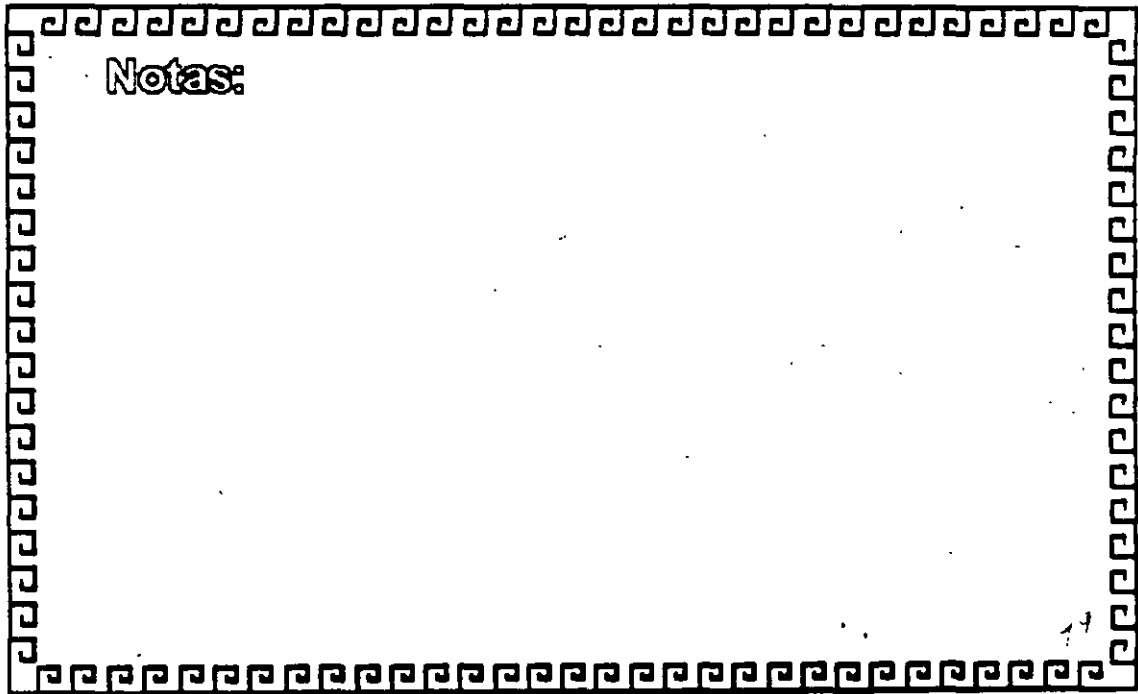
Notas:



ARQUITECTURA

Protocolos a nivel de Red

IP	Internet Protocol
ICMP	Internet Control Message Protocol
ARP	Address Resolution Protocol
RARP	Reverse Address Resolution Protocol
RIP	Routing Information Protocol
EGP	External Gateway Protocol
OSPF	Open Shortest Path First

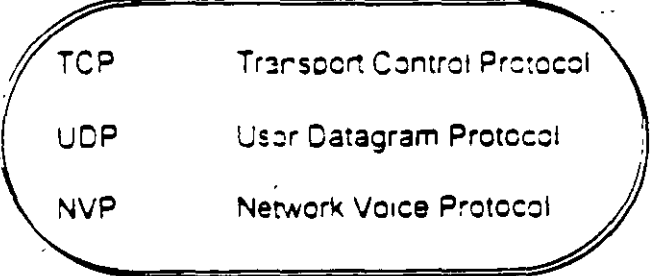


Notas:



ARQUITECTURA

Protocolos a nivel de Transporte



TCP	Transport Control Protocol
UDP	User Datagram Protocol
NVP	Network Voice Protocol



Notas:



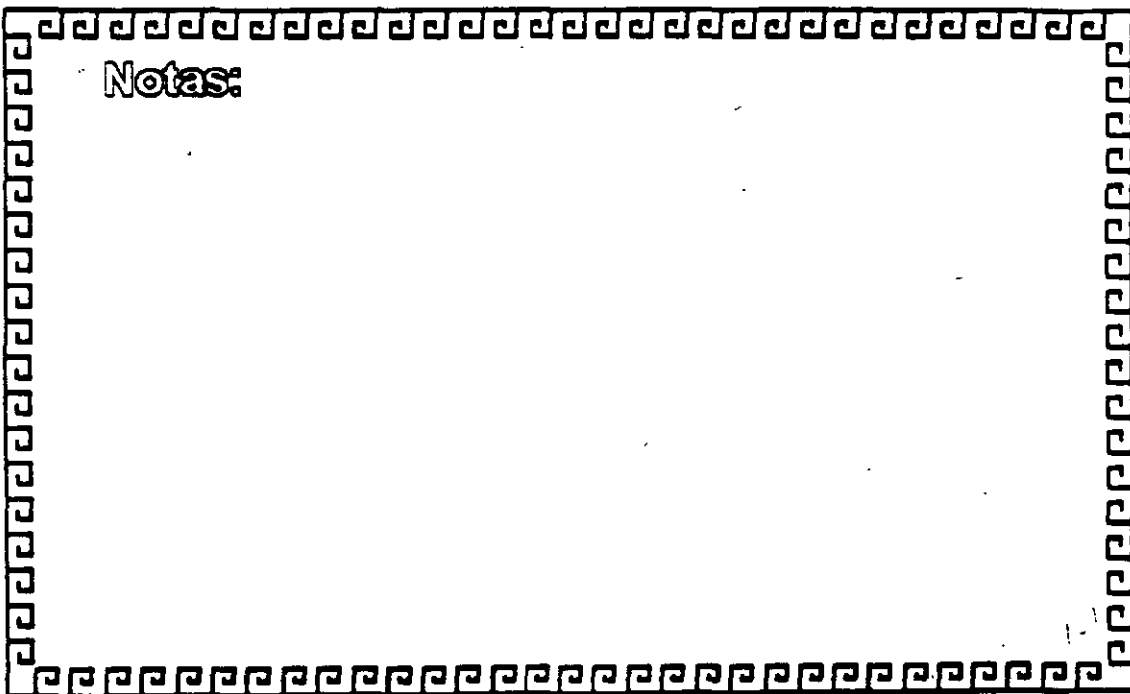


ARQUITECTURA

Protocolos a nivel de Aplicación

SMTP	Simple Mail Transfer Protocol
FTP	File Transfer Protocol
TELNET	Comunicación de Terminal
DNS	Domain Name Service
NSP	Name Service Protocol

Notas:

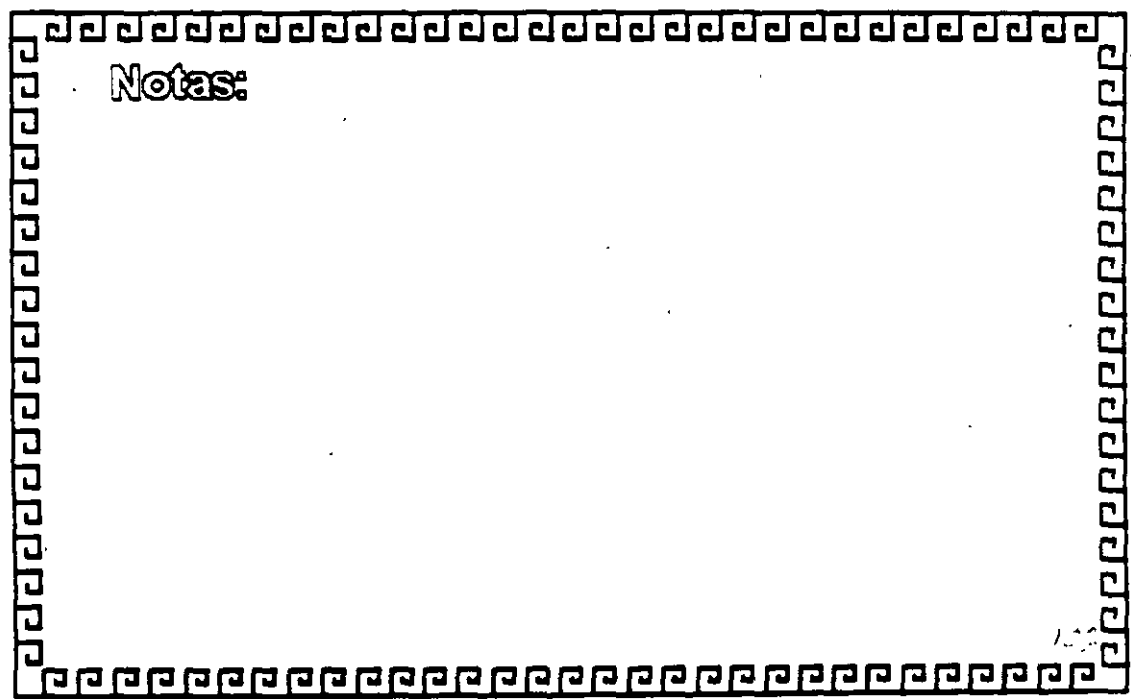




ARQUITECTURA

OSI vs TCP/IP

¿Por que TCP/IP fue un conjunto separado de estándares?



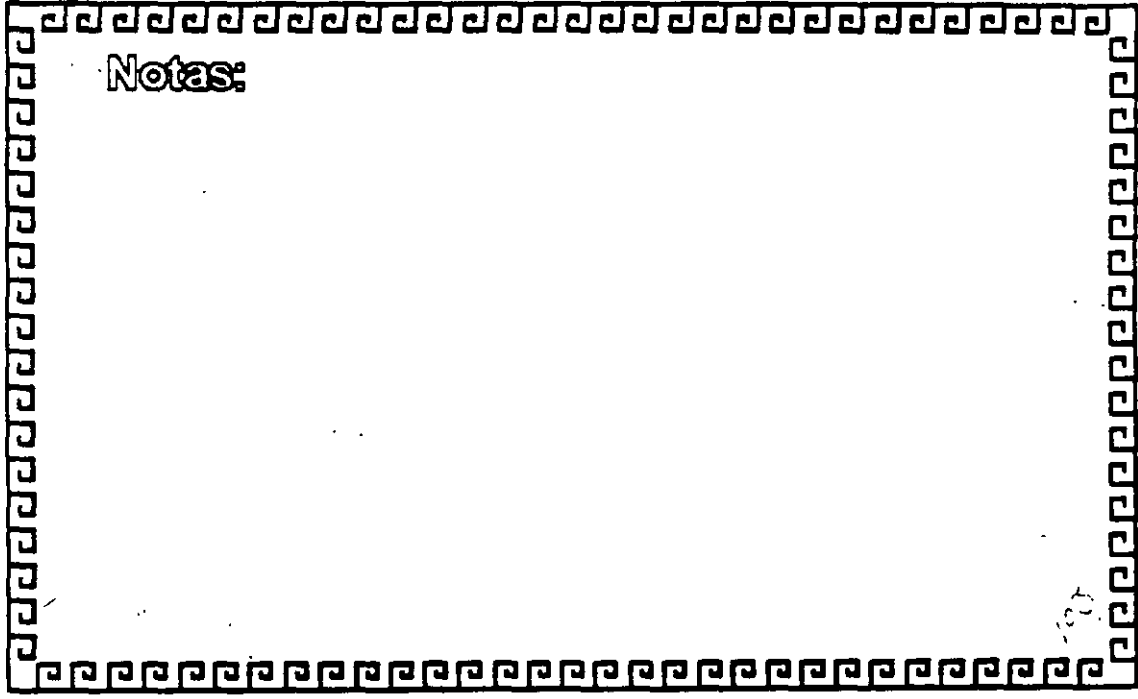
Notas:



Nivel 3. Protocolos de Red

El nivel 3 provee un fuerte poder de transmisión y otros servicios

- Entrega de paquetes punto a punto
- Amplio direccionamiento
- Identificación a varios niveles
- Fragmentación
- Datagramas de mayor envergadura
- Uso de redes con ancho de banda limitado
- Permite operación Inter-Red



Notas:



IP: Internet Protocol

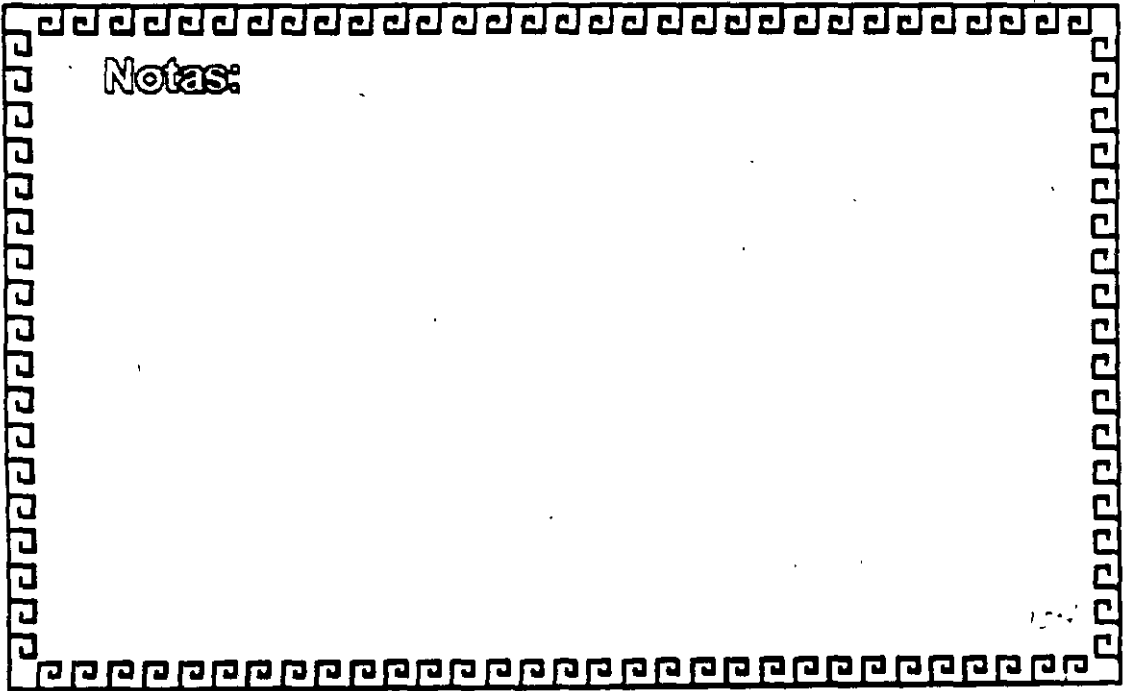
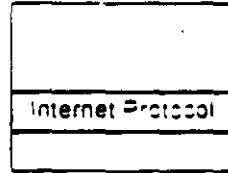
Brinda dos servicios basicos

- *Enrutamiento

- *Fragmentación/Re-ensamblaje

Utiliza direcciones IP para decidir el ruteo

Aisla los protocolos superiores de las características específicas de la red



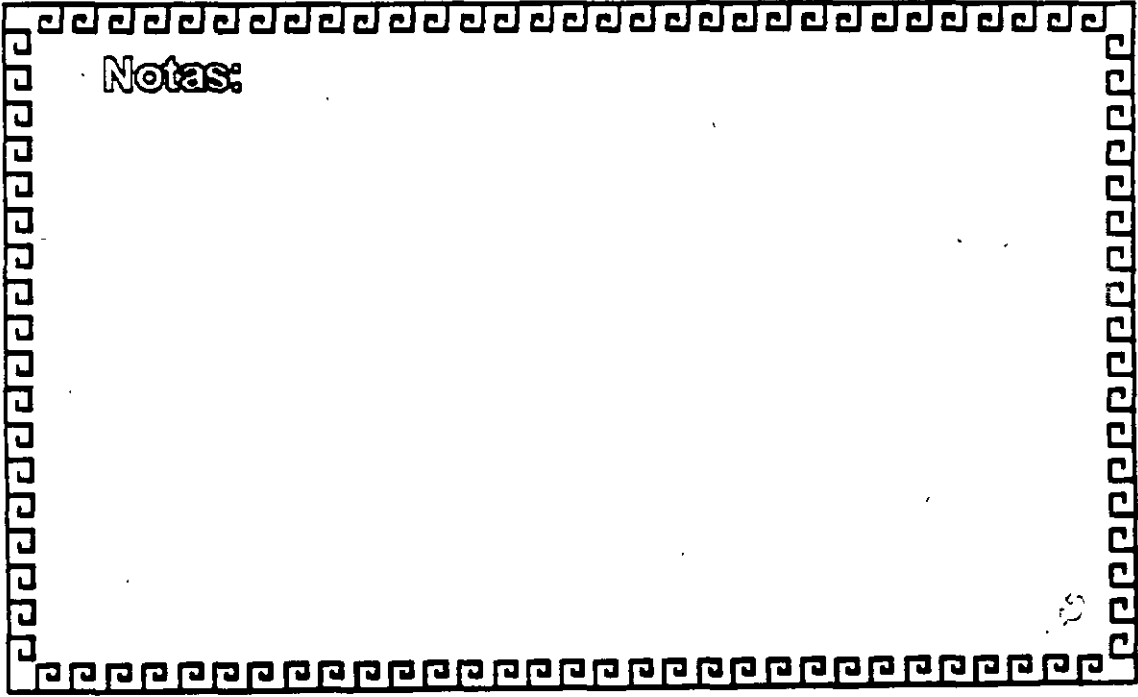
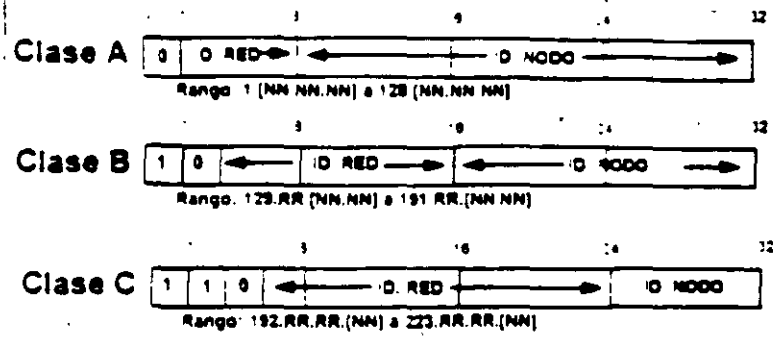
Notas:



IP: Internet Protocol

Formato de las direcciones IP

Existen 4 clases de direcciones IP A B C y D

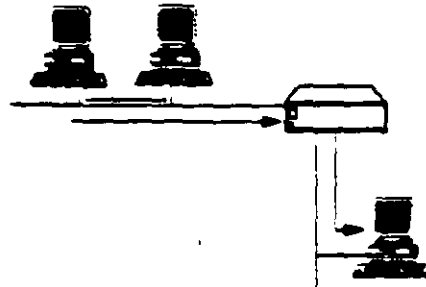


Notas:

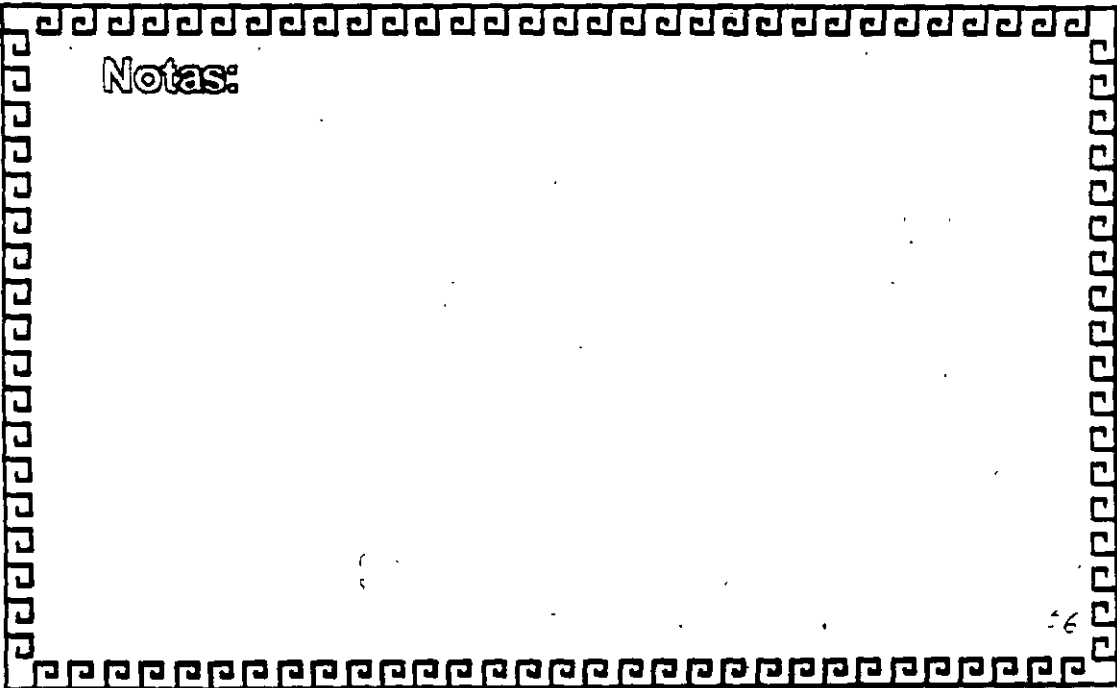


IP: Internet Protocol

Enrutamiento



Notas:



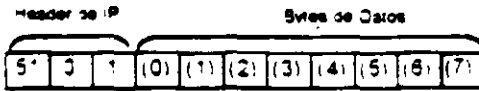


IP: Internet Protocol

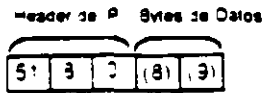
Fragmentación



Paquete de 10 Bytes



Fragmento 1



Fragmento 2

Notas:

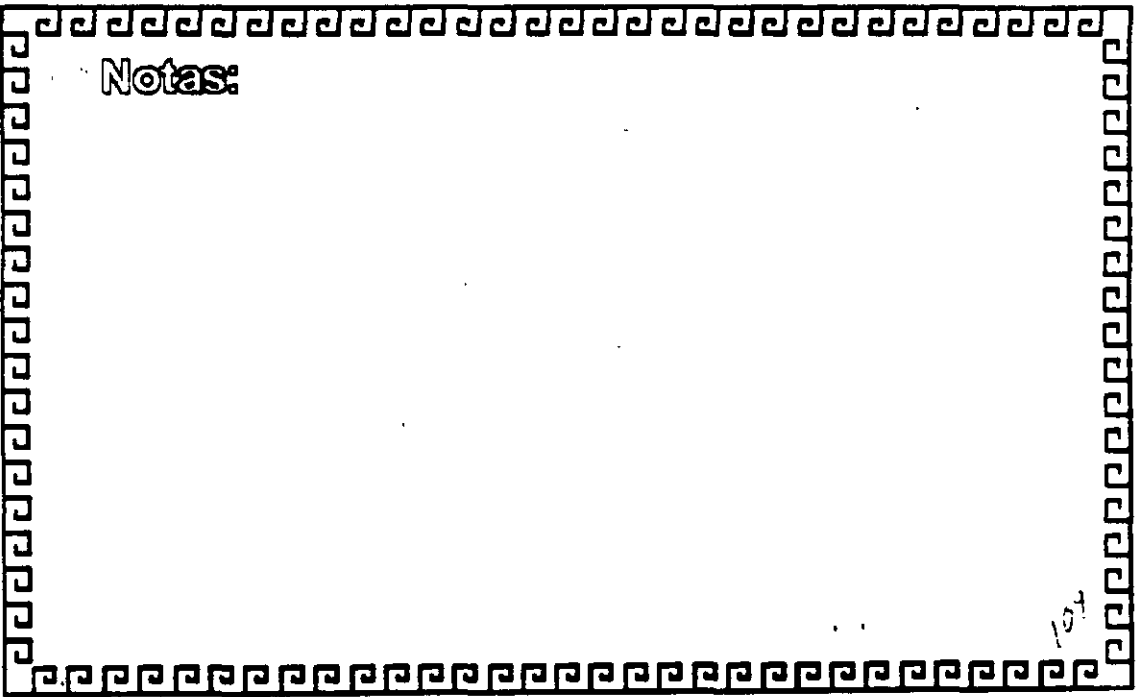


IP: Internet Protocol

Ejemplo

Nodo que recibe

- * Toma el paquete del nivel de enlace
- * Determina si el paquete ha sido fragmentado
- * Si está fragmentado lo re-ensambla
- * Pasa el datagrama a TCP



Notas:

104



ICMP

Internet Control Message Protocol:

- Brinda información de ruteo y detección de errores
- Utilizado para informar al módulo de IP acerca de
 - Paquetes que no alcanza su destino
 - Ruteadores incapaces de enviar los paquetes
 - Ruteadores que pueden enviar los paquetes por rutas mas cortas



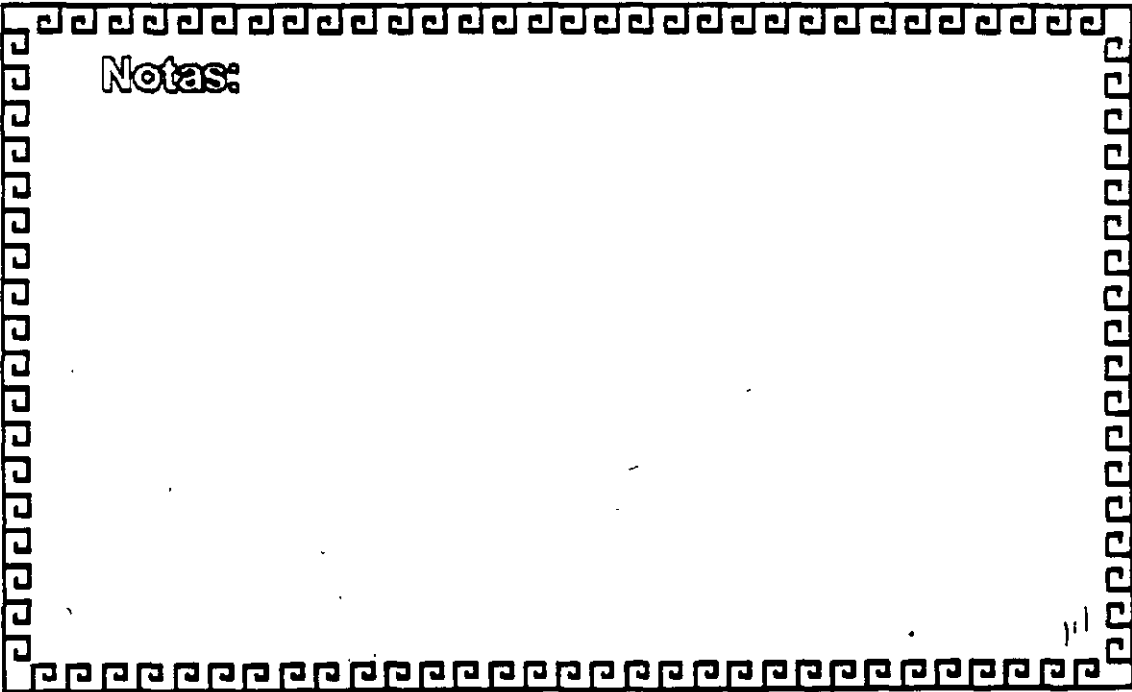
Notas:



ARP

Address Resolution Protocol

- Encuentra la dirección Ethernet (MAC) para una determinada dirección IP
- Verifica en cache local
- Emite un broadcast si la dirección no está en el cache



Notas:

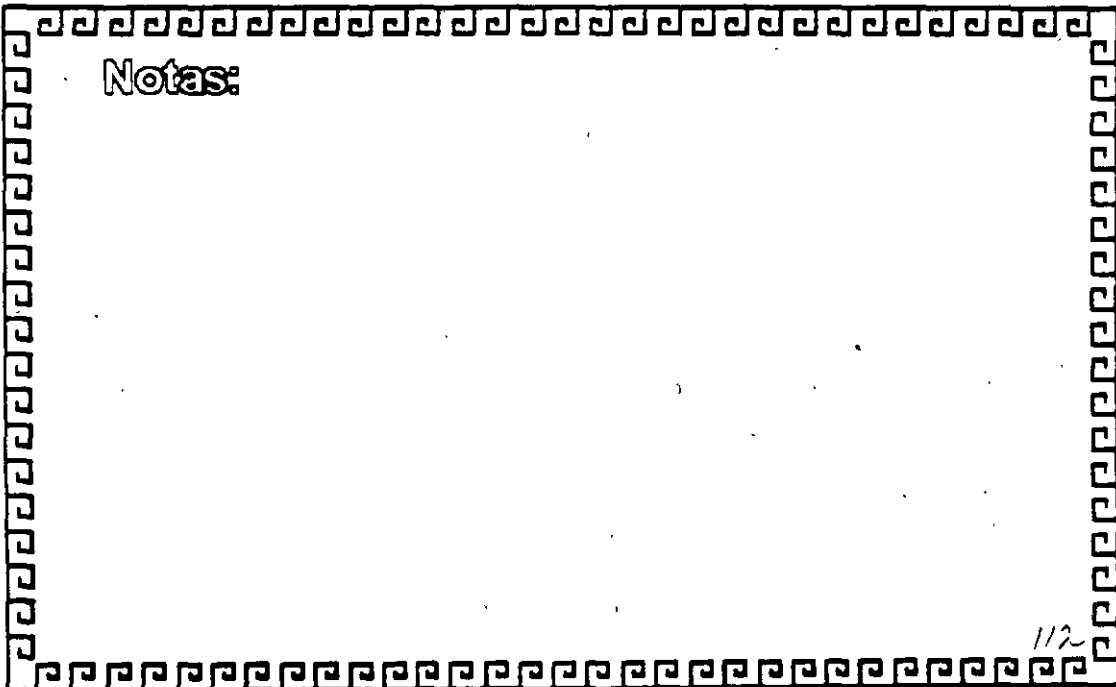


RARP

Reverse Address Resolution Protocol

* Encuentra la dirección IP para una determinada dirección Ethernet (MAC).

- * La operación se realiza a través de un broadcast
- * Se requiere de un proceso de servidor RARP por cada red



Notas:

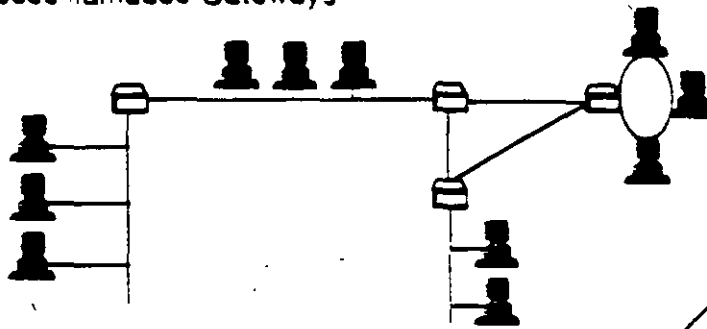
112



IP: Internet Protocol

Direccionamiento Inter-Red

* Una Inter-red (internet) esta formada por una colección de redes individuales, unidas por ruteadores, a veces llamados Gateways



Notas:

113

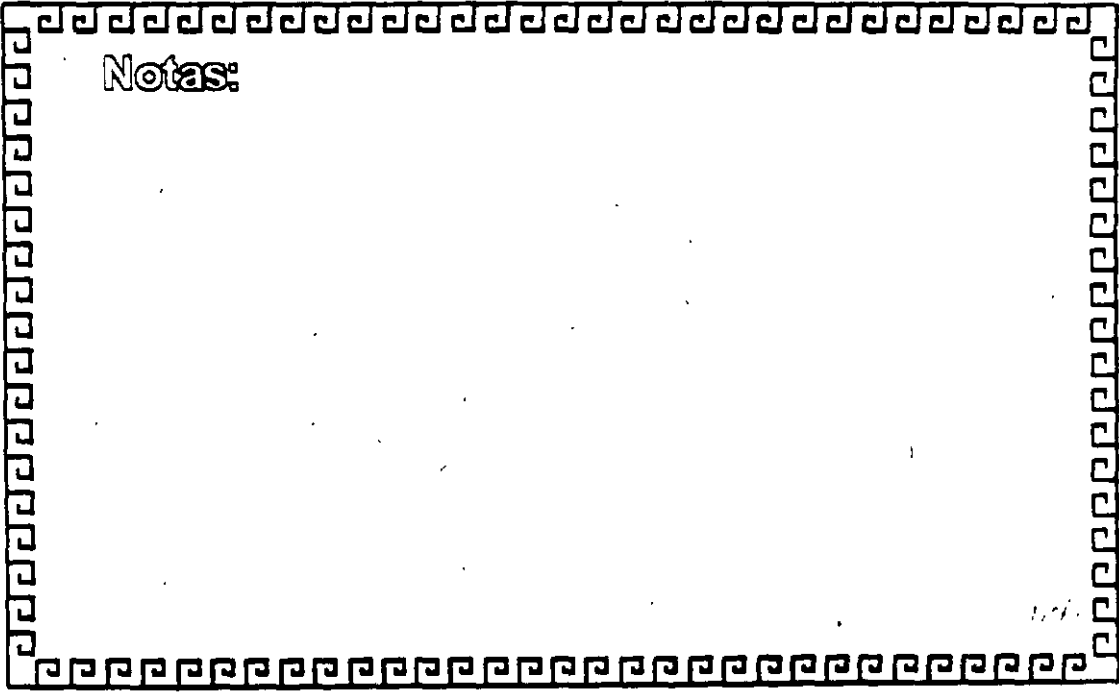


Ruteo de IP

Protocolos de ruteo

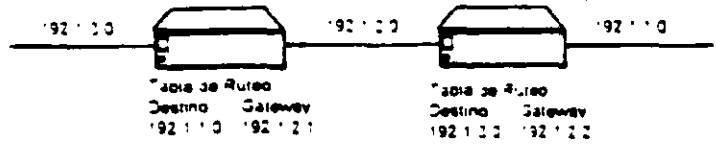
* Intercambiar entre dispositivos (generalmente routers) información acerca de las redes que conectan

- * Protocolos Intradominio
- * Protocolos Interdominio



Notas:

Ruteo de IP



Notas:



Protocolo RIP

Routing Information Protocol

- * Es un protocolo de ruteo simple
- * Cada ruteador transmite el costo y la direccion destino a sus vecinos
- * Existen dos tipos de paquetes
Solicitud (request)
Respuesta (Response)



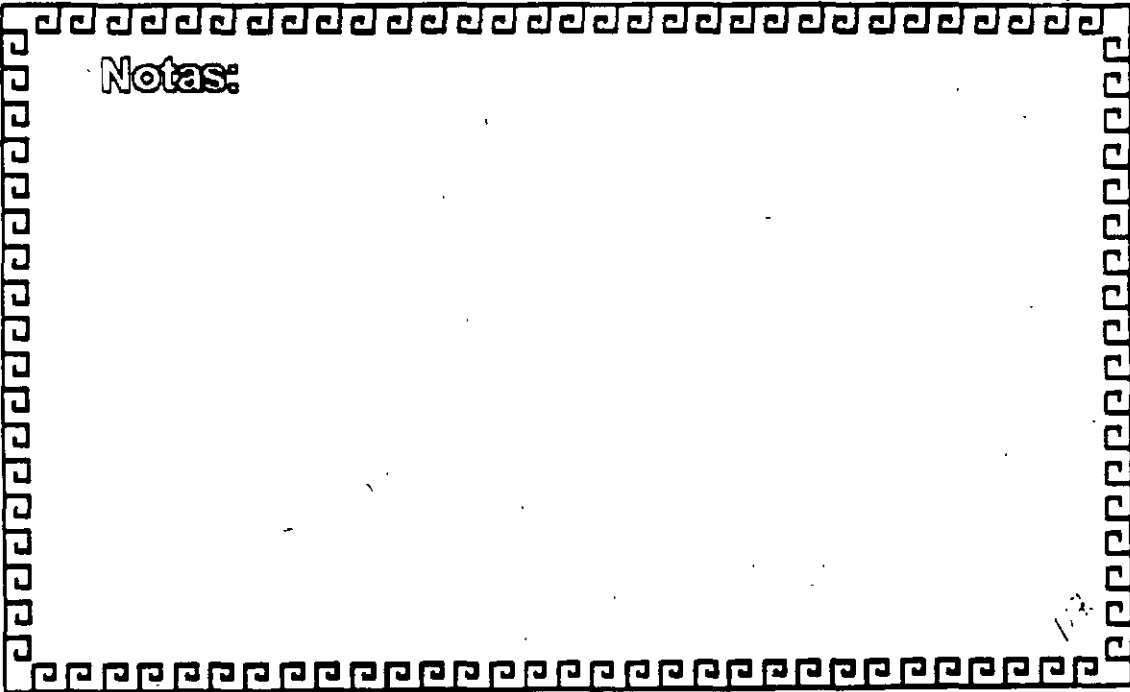
Notas:



Protocolo RIP

Formato de paquete

comando
sign
reservado
identificación de dirección de familia
Dirección
metrca




Notas:

1/2



Protocolo OSPF

Open Shortest Path First

- * Protocolo no propietario
 - * Divide las redes en
 - Area
 - Sistema Autonomo
 - Sistema Global
 - * Sistema de seguridad para la propagación de las rutas disponibles
 - * Uso de diferentes tipos de métricas .
- 

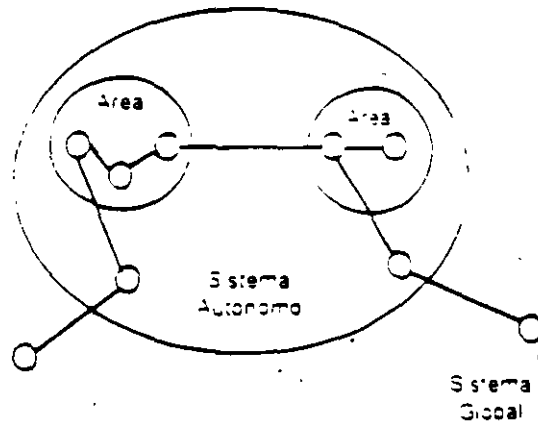


Notas:



Protocolo OSPF

Divisiones o particiones



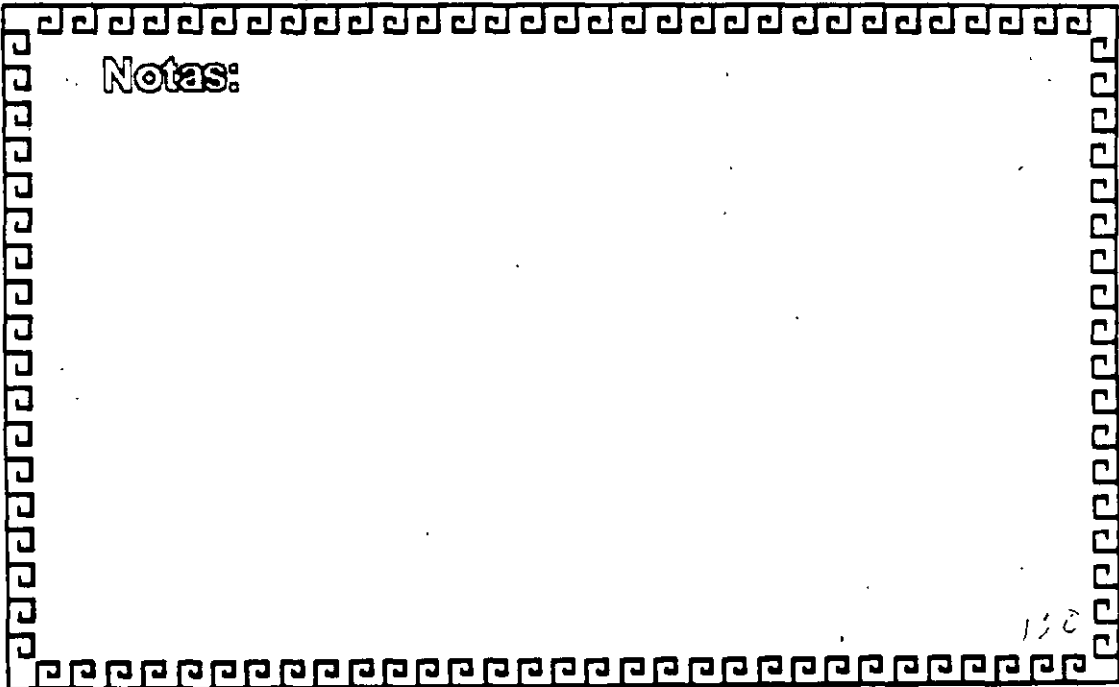
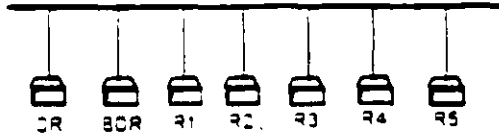
Notas:



Protocolo OSPF

Propagacion de las rutas

Area







Protocolo OSPF

Tipos de paquetes de propagación de rutas

- * Tipo 1. Routing Link Advertisement
- * Tipo 2. Network Links Advertisement
- * Tipo 3. Network Summary Link Advertisement
- * Tipo 4. AS Boundary Router summary Link Advertisement
- * Tipo 5. AS External Link Advertisement



Notas:




Protocolo EGP

External Gateway Protocol

- Fue uno de los primeros protocolos
- Tres funciones o aspectos
 - Adquisición de Vecinos
 - Confirmación de Vecinos
 - Información de Ruteo
- No funciona con enlaces redundantes
- No se incluye métrica en la información
- Permite que en una lan un sólo router transmita la información a ei/los routers de otros dominios



Notas:



Nivel 4. Protocolos de Transporte

El nivel de transporte provee a una máquina con conexiones punto a punto independientes de la subred y servicios de transacción

- Provee enlaces confiables y eficientes entre procesos
- Forma en conjunto con los niveles inferiores una robusta plataforma de comunicaciones
- Realiza los enlaces virtuales
- Tiene dos protocolos principales
 - TCP
 - UDP



Notas:




Protocolo TCP

Transmission Control Protocol

Asignación de números de puerto para transmisión de datos
Reconocimiento de Datos recibidos
Regulación del flujo de datos
División de los mensajes en datagramas
Verificación de los datagramas

Administración
Establecimiento
Mantenimiento
Terminación



Notas:

124



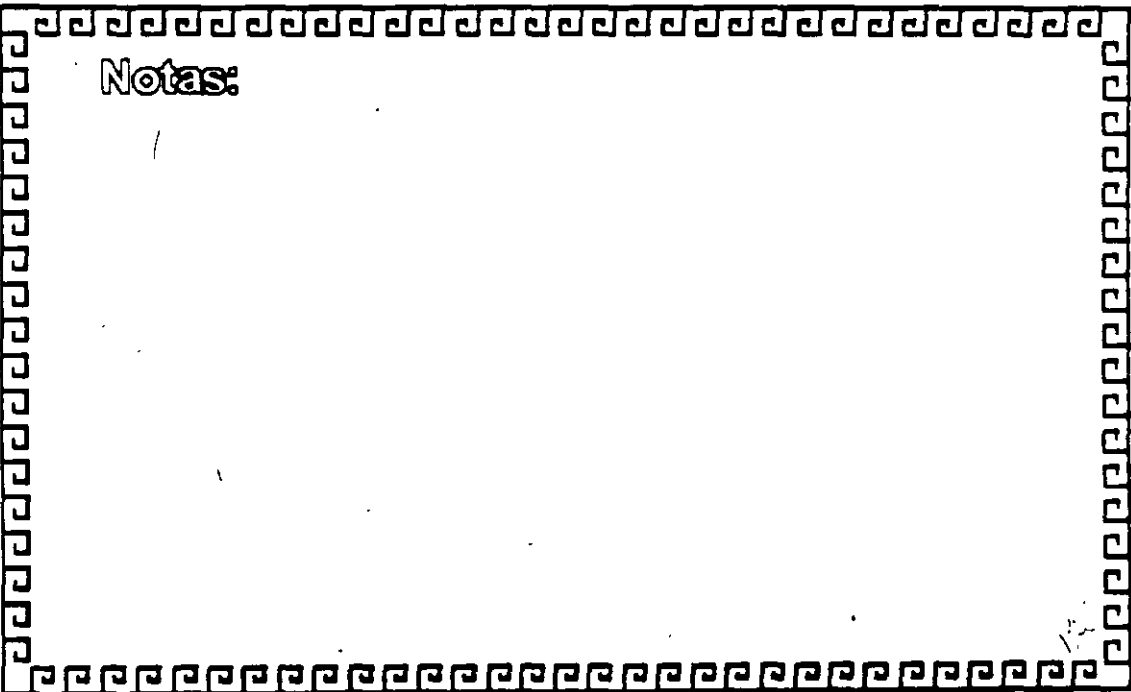
Protocolo UDP

User Datagram Protocol

UDP brinda servicio de datagramas a los programas del usuario
No garantiza una transferencia confiable de los datos
Envia/Recibe datos sin capacidad de retransmision
Supone que la aplicacion de mas alto nivel realiza la validación

Utilizado por:

NFS (Network File System)
SNMP
TFTP



Notas:

Protocolo NVP

Network Voice Protocol

Servicio para transporte de voz digitalizada

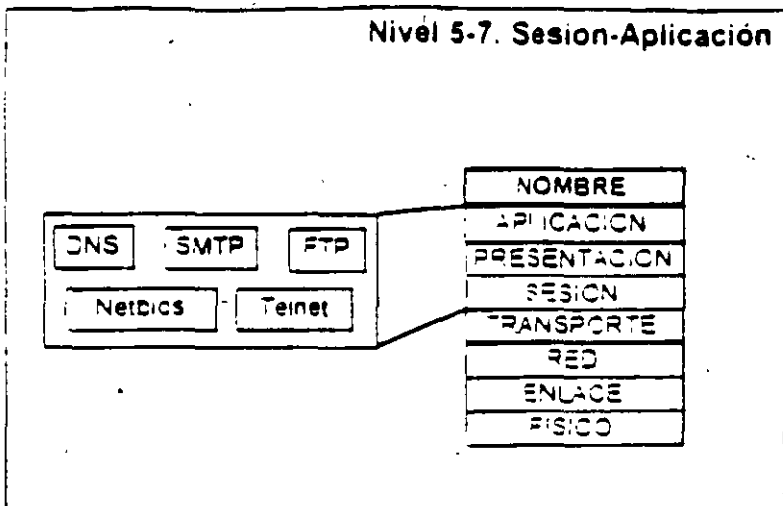
Protocolo de transacción de tiempo real

Utiliza IP para transmitir información

Empieza algoritmos de compresión

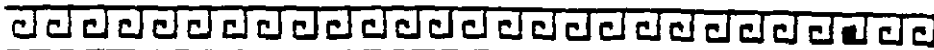
Es conexión less

Notas:



Notas:

12/1

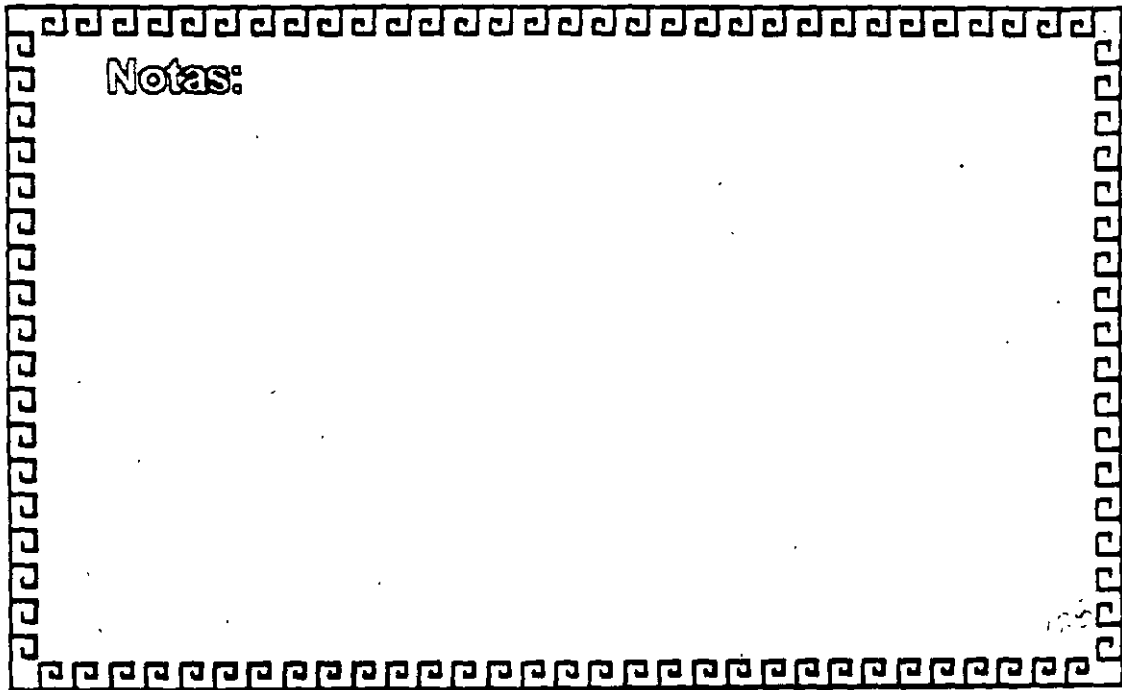


Nivel 5-7. Sesión-Aplicación

Nivel de Sesión

Nivel de Presentación

Nivel de Aplicación



Notas:

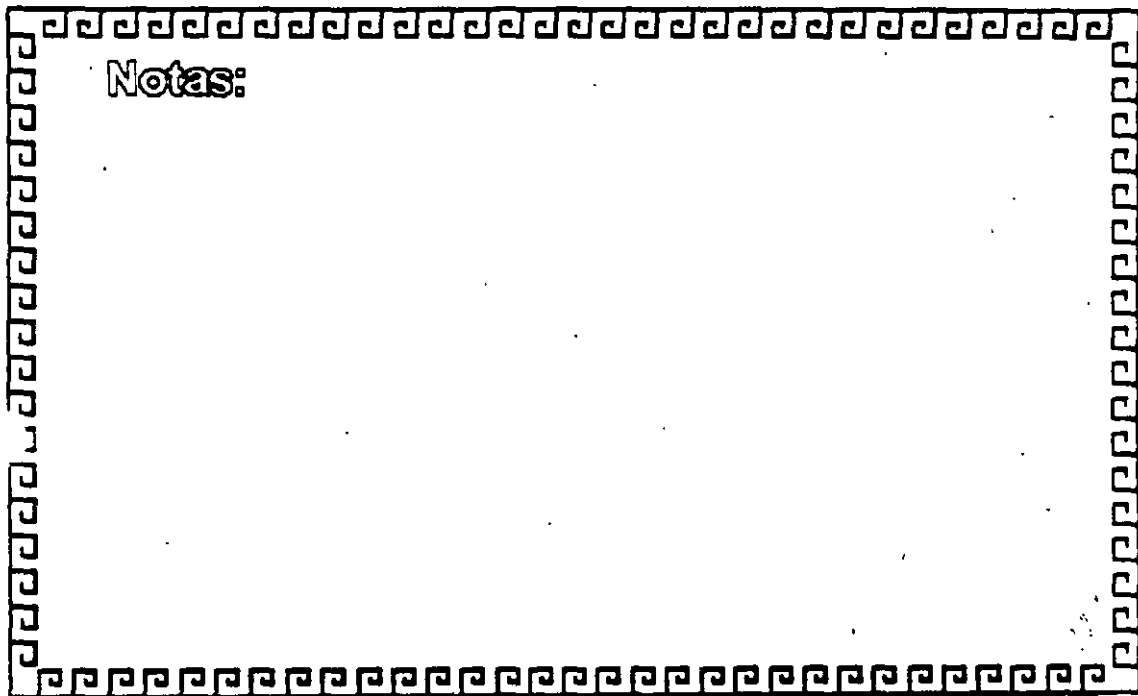


Protocolo Telnet

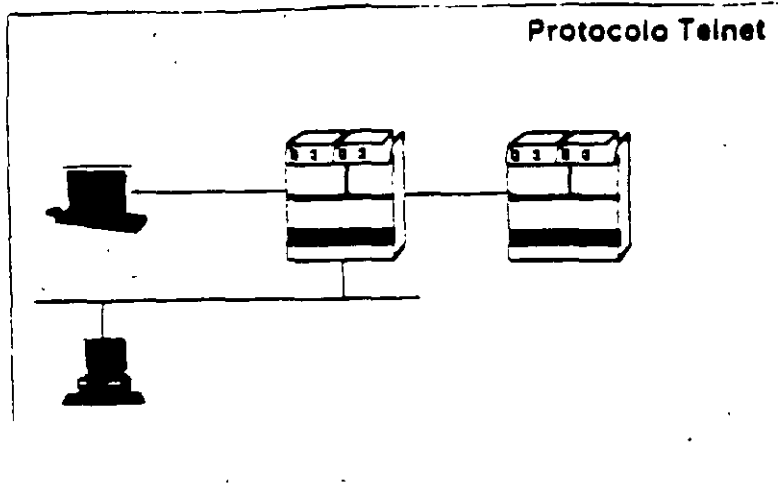
Protocolo de Acceso Remoto e Interactivo de terminal

Brinda una conexión virtual a nodos remotos

Permite a los usuarios acceder nodos remotos como si fueran terminales. Físicamente Conectados a el Host



Notas:



Notas:

130



Protocolo Telnet

Ejemplo

```
Telnet>open apollo1
Trying open
Connected to apollo1
Escape Character is ^]

Username
Password:
```

Notas:

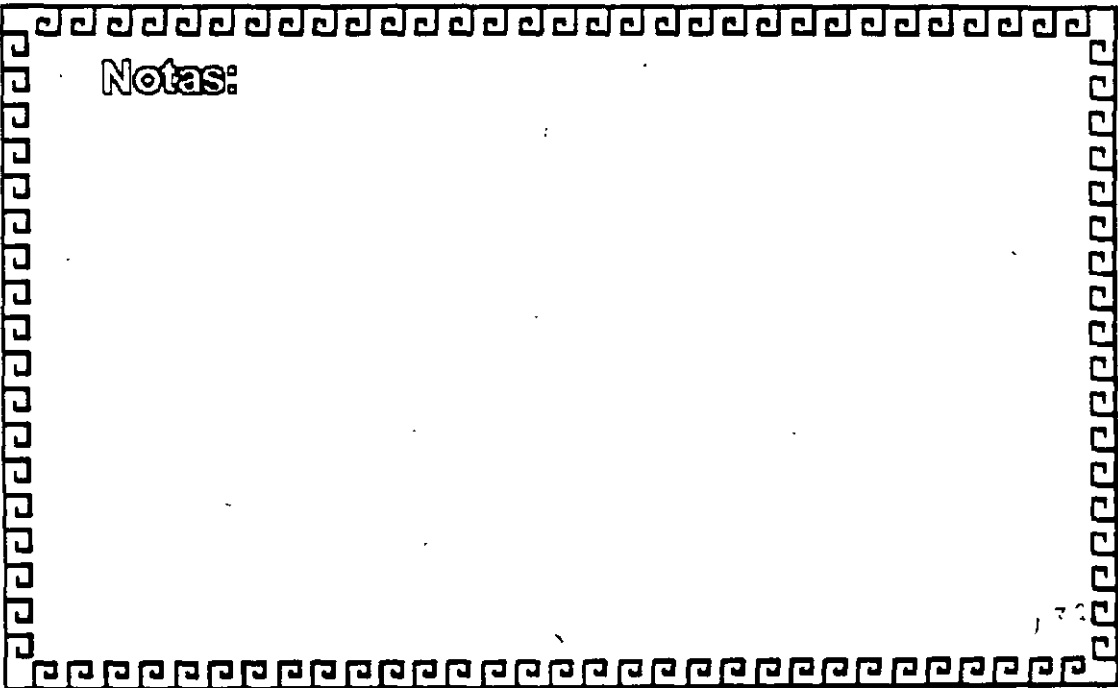


Protocolo Telnet

Comandos

open	Conectarse a un host
close	Cerrar sesion actual
escape	Definir caracter de escape
exit	Fin de telnet
local echo	Cambio de eco (Encendido, apagado)
status	Información de cada sesión
?	Ayuda

Notas:

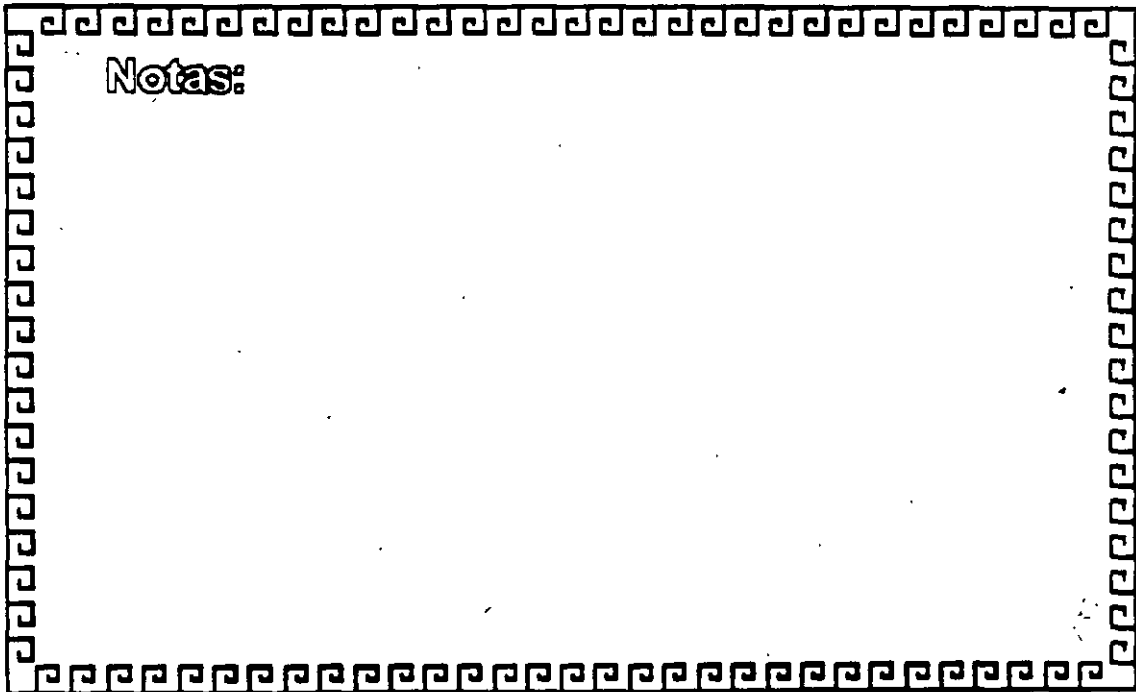




Protocolo FTP

File Transfer Protocol

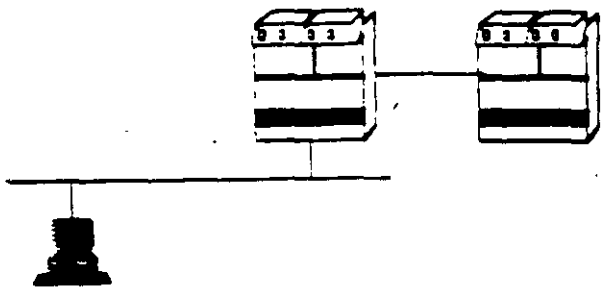
- FTP permite el envío y recepción de uno o mas archivos en forma interactiva
- Soporta formatos de archivo en ASCII, Binario y EBCDIC
- Modo de transmisión 'Stream' Bloque o comprimido
- Permite las manipulaciones sencillas dentro de los sistemas de archivos Locales y Remotos



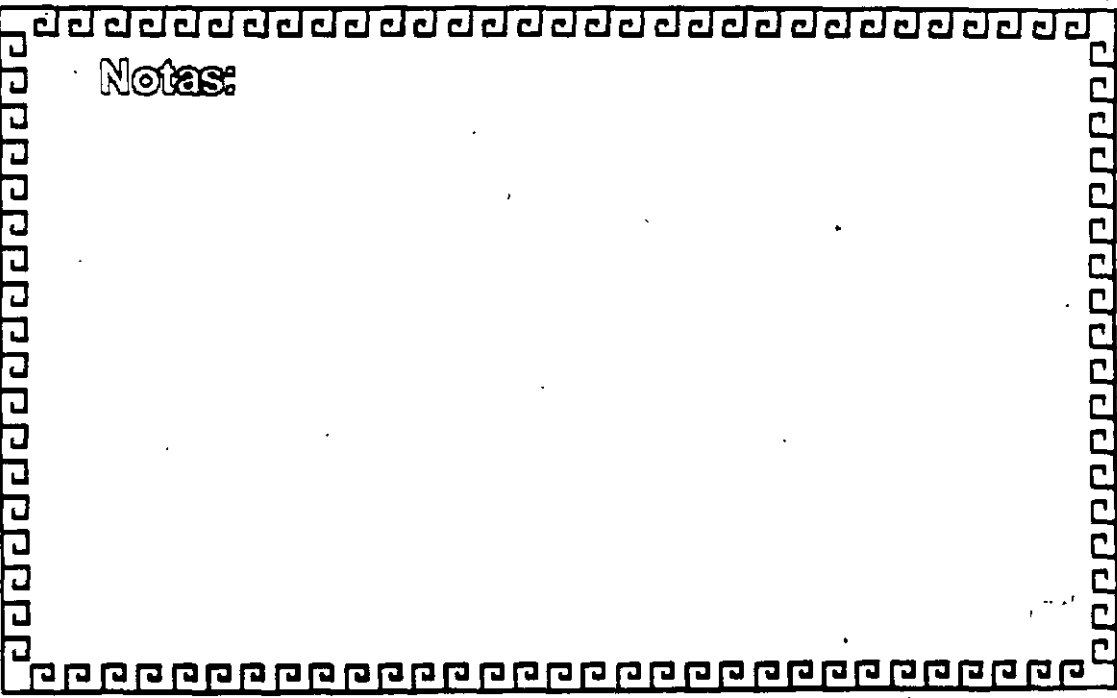
Notas:



Protocolo FTP



Notas:






Protocolo FTP

Ejemplo

```
$fto
* open vax1
Name.
<Enter PASS Comand)
Password
<User logged inn default directory)
*
*get remotefile local file
*put local file mewfilename
bye
$
```



Notas:



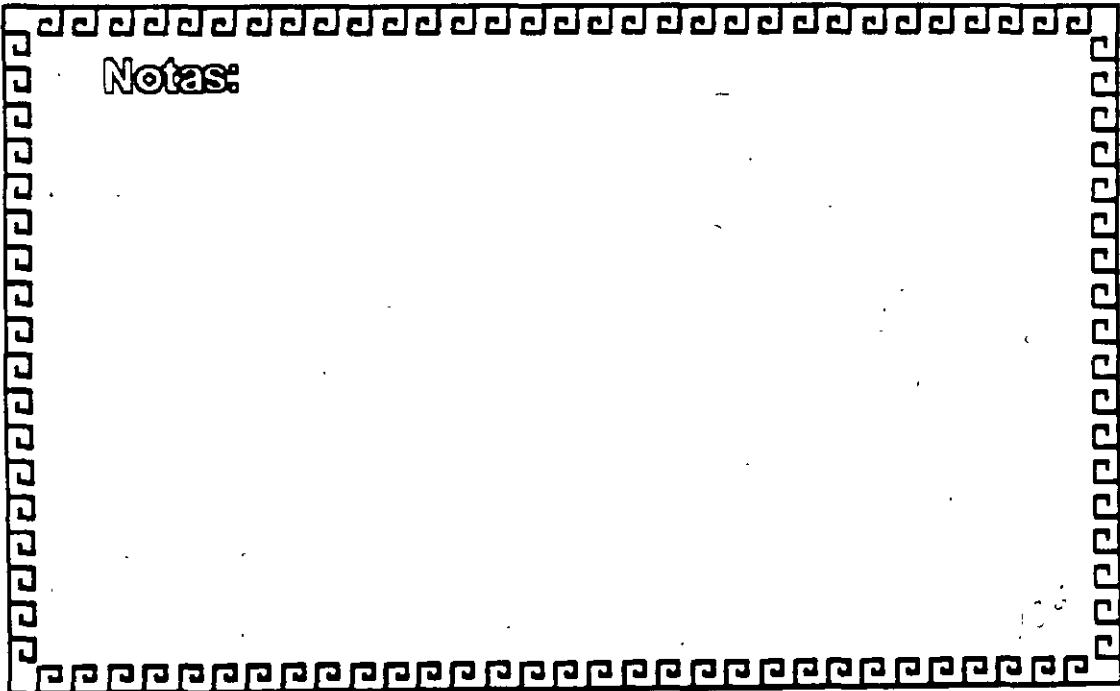
TCP/IP

PROTOCOLO FTP

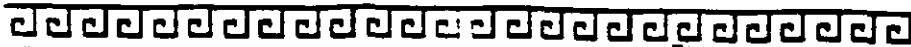


Comandos

Abort	Interrupcion
Ascii	Define modo de transferencia a ascii
Bget	Leer un archivo en modo binario
Bput	Enviar un archivo en modo binario
Bell	Alarma para indicar fin de transferencia
Bye	Termina enlace y sale
Case	Hace el cambio de los nombres de archivos locales a minusculas
Cd	Cambio de directorio remoto



Notas:



TCP/IP

PROTOCOLO FTP

COMANDOS	
Chdirup	Cambio de directorio en host remoto al inmediato superior
Comand file	Ejecuta los comandos de ftp en batch
Directory	Desplegar contenido
Delete	Borrar remoto
Disconnect	Terminar sesión
File	Enviar la estructura de un archivo



Notas:



Protocolo FTP		
Usuario	Programa Cliente	Programa Server
Site	>User Control Process	Server Control
* open host	>Control connector Control connector	<Server replies <FTP available
*login tom	>Control Connection Control Connector	Server accepts <&Request password
password	>Control Connection Control Connection	<Server accepts <Server accepts
hash User Creates		Server Creates
		Data process
*get <filename>	Data connection Data connection	<Server opens <Server sends data S



Notas:



Protocolo FTP

<u>Usuario</u>	<u>Programa Cliente</u>	<u>Programa Server</u>
ACK each segment>	Data connection Data connection	Server sends <last data byte
ACK received OK>	Data connection Data connection	<Server closes Data connection
ACK>	Data connection Control Connection	<Server Data Process terminates
Data proc. Term> File transfer complete	Control connection	




Notas:



Protocolo NFS

Network File System

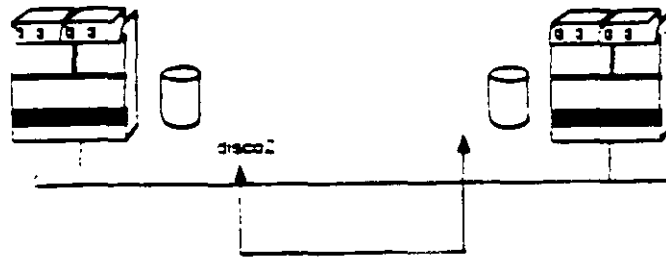
- Originado y popularizado por SUN Microsystems
 - Diseñado para ser portado facilmente a diferentes sistemas operativos
 - Brinda acceso transparente a sistemas remoto de archivos
 - Los usuarios no necesitan saber la localidad física de los discos
- 



Notas:



Protocolo NFS




Acceso a discos remotos en forma transparente

Notas:



Protocolo SMTP

Simple Mail Transfer Protocol

- * Un de los protocolos mas implementados
 - * Define como transmitir mensajes entre 2 usuarios
 - * Se basa en Spooling para el envio de Mensajes
 - * Se conoce como envio de mensajes punto a punto
 - * Describe la estructura del mensaje y especifica el protocolo para el intercambio de correo
- 



Notas:



Servicios de Nombramiento

HOSTS

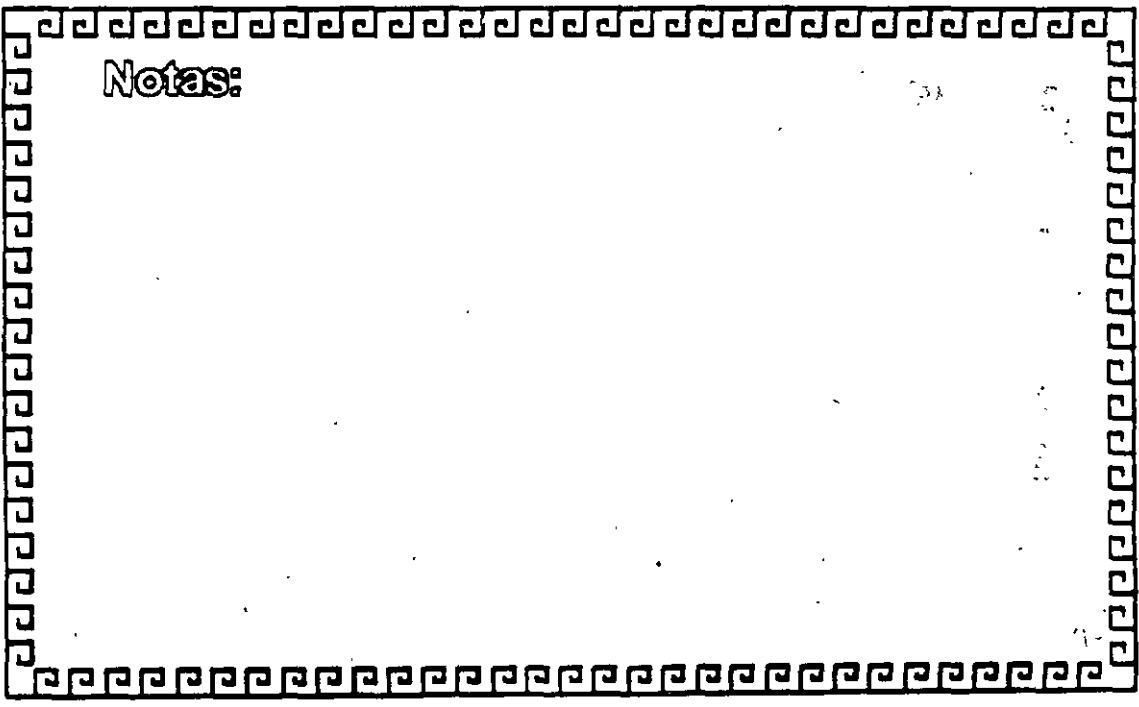
Contiene relacion de nombres y direcciones IP sobre cada nodo de la red

Name Service

Un servicio central de nombramiento. El archivo de nombres en el servidor es similar al archivo de "HOSTS"

Domain Name Service

Sistema descentralizado de nombramiento
Utiliza varios archivos para resolver las direcciones de IP
Especificacion RFC 1032-1034



Notas:



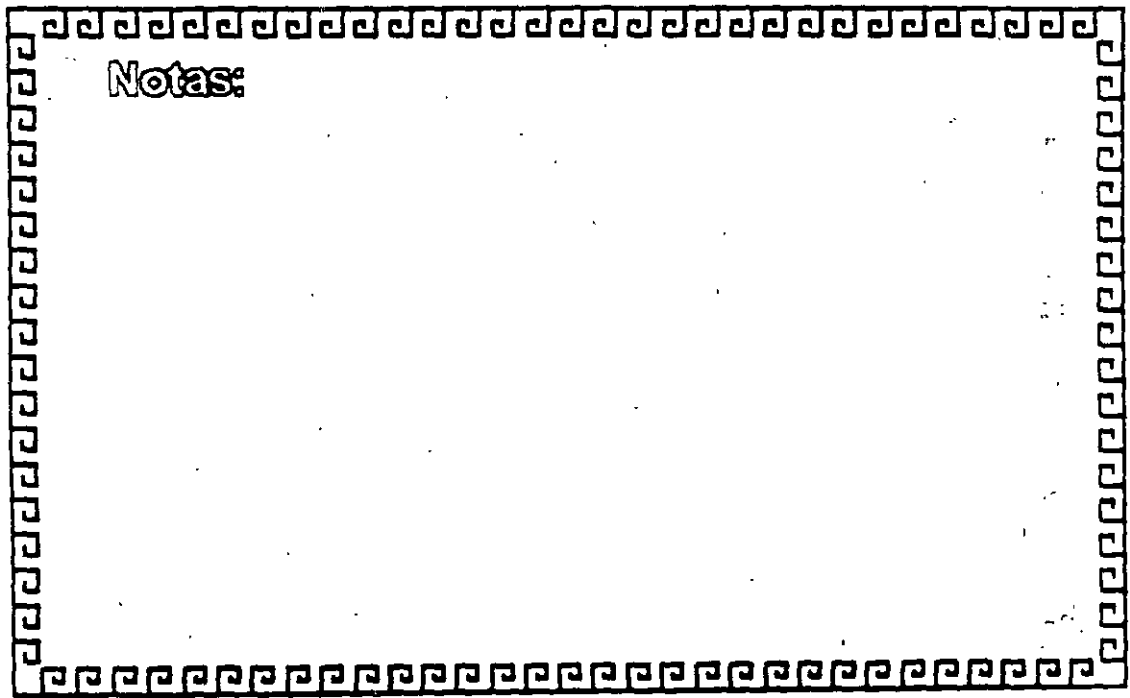
TCP/IP

Servicios de Nomenclamiento

Archivo HOSTS	
127.1.1.1	Localhost
128.90.1.3	Vax1
128.90.1.9	Sun
128.90.3.9	Vax3
192.1.10.25	Apollo1
192.1.10.95	Hp9000 conta1
192.1.10.93	Hp9001 conta2



Notas:



DNS

Domain name Service

Protocolo de nombramiento

Brinda traducción de nombre-dirección IP

Dominio: Grupo de Hosts

"Domain Name Server"

Notas:



Domain Name Service

Información del servidor

Dirección Internet

Tipo de computadora

Lista de servicios brindado por computadoras

Servidor

Servidores Maestros

Primario

Secundario



Notas:



TCP/IP Sobre X.25

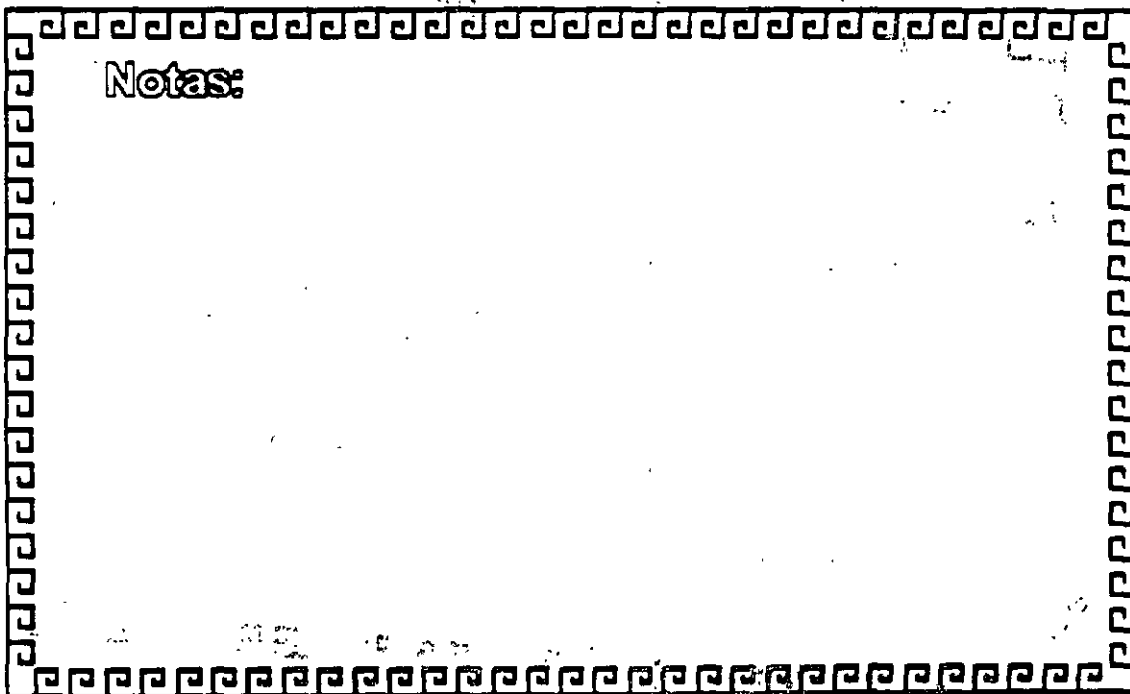
Implementacion de TCP/IP para redes de area amplia

2 opciones interno o externo

Generalmente con conexiones dinamicas

En caso de no usar la linea esta se desconecta temporalmente

La fragmentación la realiza X 25



Notas:

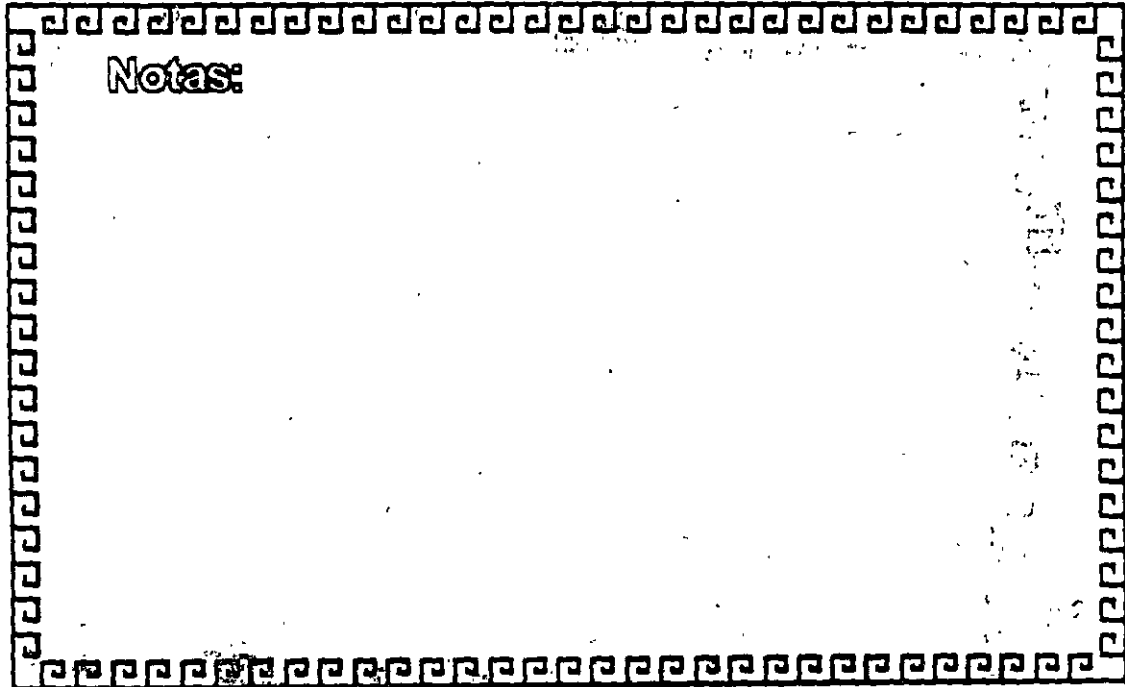


TCP/IP Sobre X.25

Se requiere tablas de conversion para determinar equivalencias entre X.25 y TCP/IP

Ejemplo

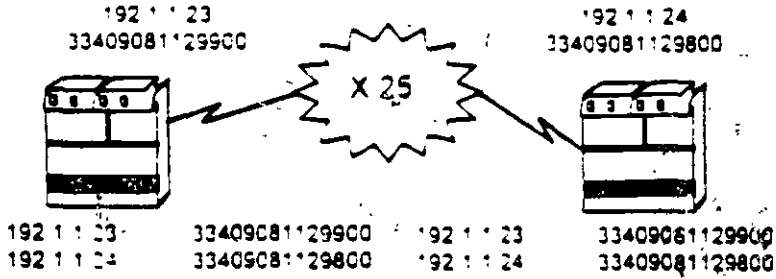
192 1 20 3	33409081129900
192 1 21 5	38219087113800



Notas:



TCP/IP Sobre X.25



Notas: