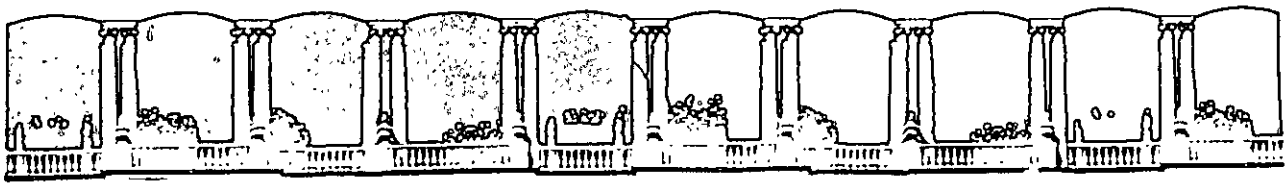


FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

CENTRO DE INFORMACION Y DOCUMENTACION
"ING. BRUNO MASCANZONI"

El Centro de Información y Documentación Ing. Bruno Mascanzoni tiene por objetivo satisfacer las necesidades de actualización y proporcionar una adecuada información que permita a los ingenieros, profesores y alumnos estar al tanto del estado actual del conocimiento sobre temas específicos, enfatizando las investigaciones de vanguardia de los campos de la ingeniería, tanto nacionales como extranjeras.

Es por ello que se pone a disposición de los asistentes a los cursos de la DECFI, así como del público en general los siguientes servicios:

- **Préstamo interno.**
- **Préstamo externo.**
- **Préstamo interbibliotecario.**
- **Servicio de fotocopiado.**
- **Consulta a los bancos de datos: librunam, seriunam en cd-rom.**

Los materiales a disposición son:

- **Libros.**
- **Tesis de posgrado.**
- **Noticias técnicas.**
- **Publicaciones periódicas.**
- **Publicaciones de la Academia Mexicana de Ingeniería.**
- **Notas de los cursos que se han impartido de 1980 a la fecha.**

En las áreas de ingeniería industrial, civil, electrónica, ciencias de la tierra, computación y, mecánica y eléctrica.

El CID se encuentra ubicado en el mezzanine del Palacio de Minería, lado oriente.

El horario de servicio es de 10:00 a 19:30 horas de lunes a viernes.

FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA
A LOS ASISTENTES A LOS CURSOS

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

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

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

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

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

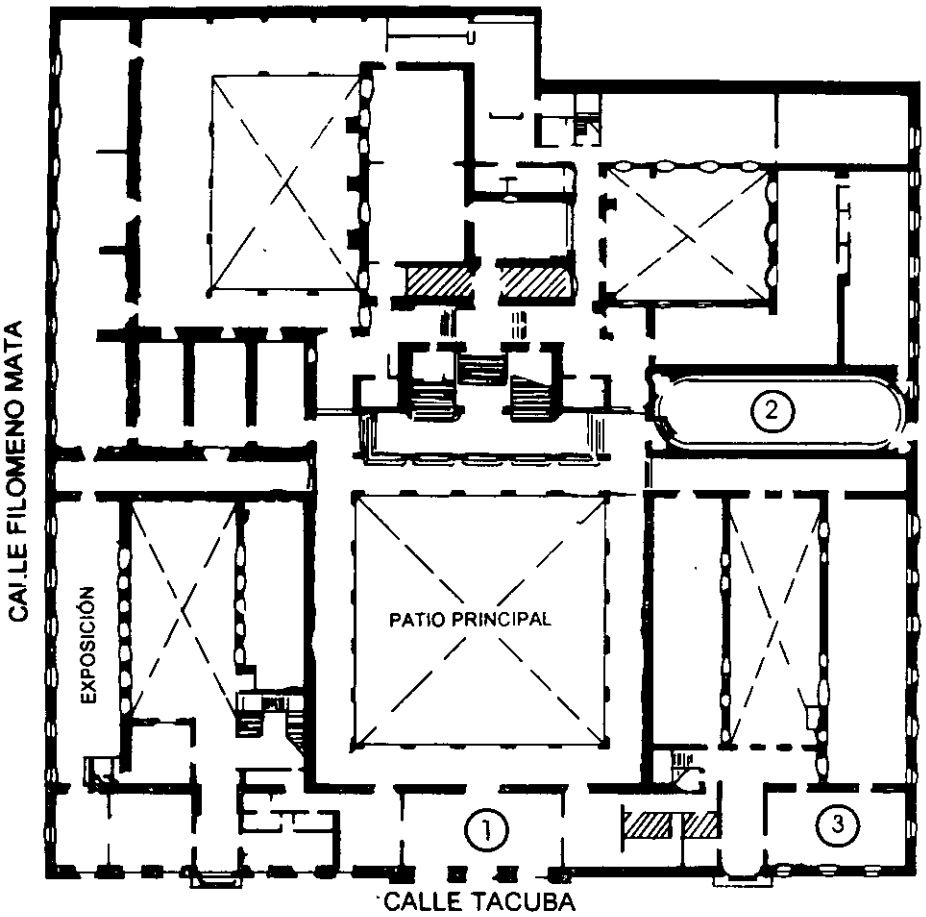
Con el objeto de mejorar los servicios que la División de Educación Continua ofrece, al final del curso deberán entregar la evaluación a través de un cuestionario diseñado para emitir juicios anónimos.

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

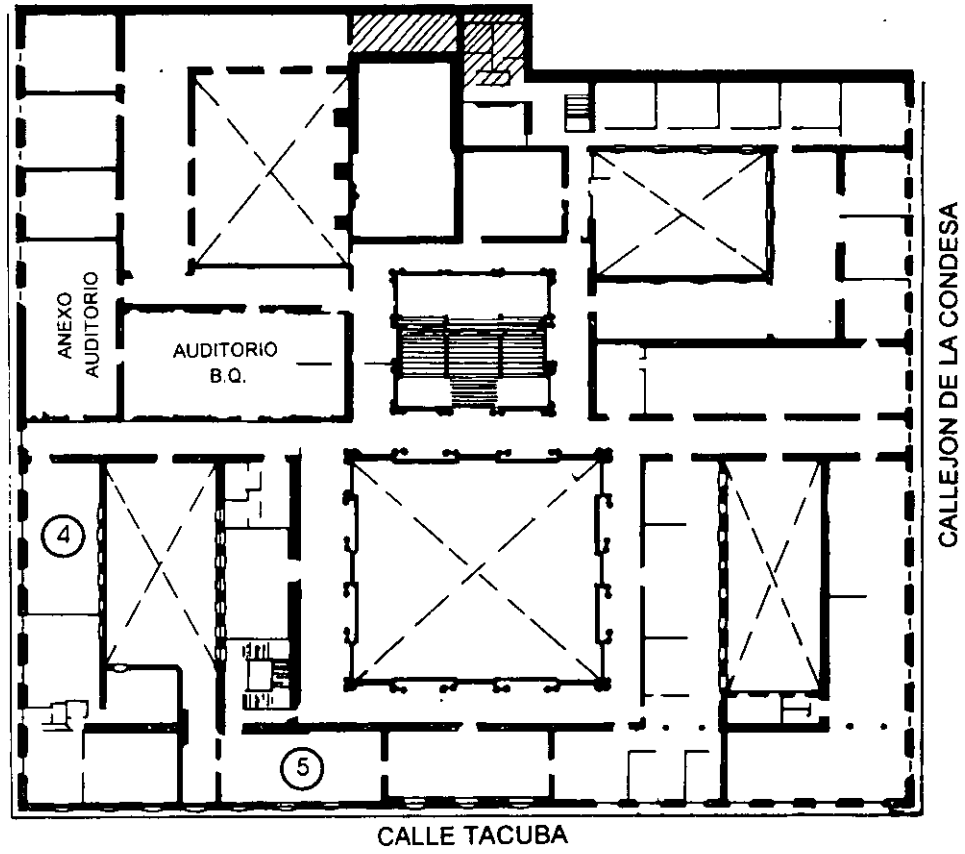
Atentamente
División de Educación Continua.

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

PALACIO DE MINERIA

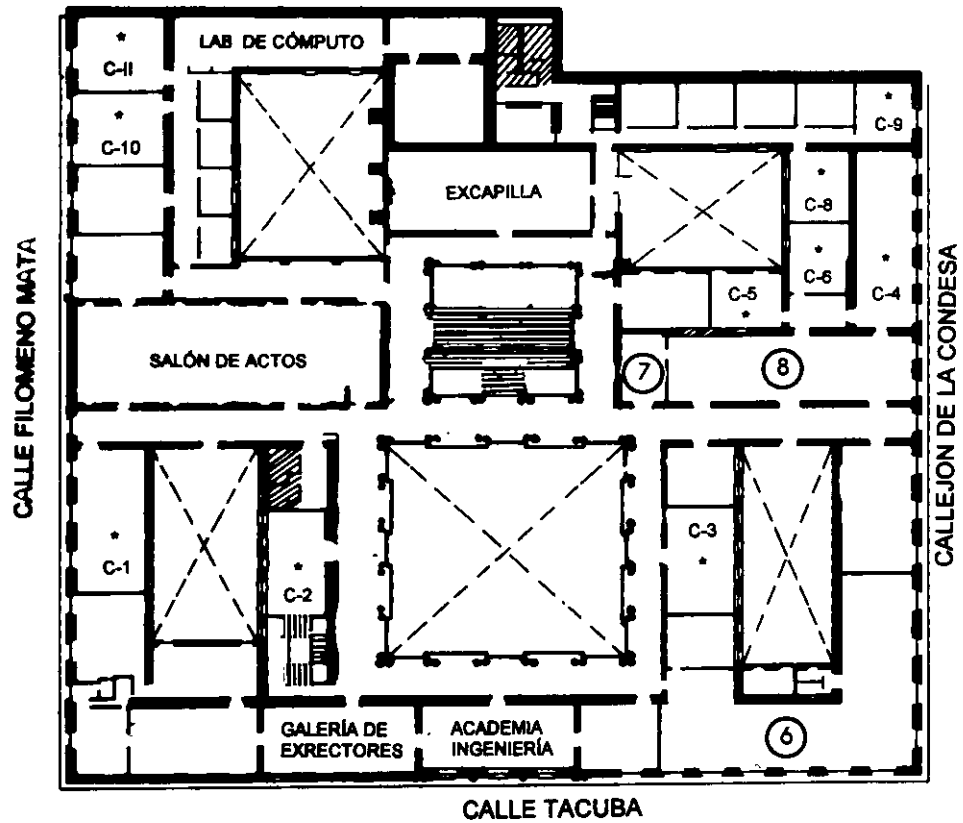


PLANTA BAJA



MEZZANINNE

PALACIO DE MINERIA



1er. PISO

GUÍA DE LOCALIZACIÓN

1. ACCESO
2. BIBLIOTECA HISTÓRICA
3. LIBRERÍA UNAM
4. CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN "ING. BRUNO MASCANZONI"
5. PROGRAMA DE APOYO A LA TITULACIÓN
6. OFICINAS GENERALES
7. ENTREGA DE MATERIAL Y CONTROL DE ASISTENCIA
8. SALA DE DESCANSO

SANITARIOS

* AULAS

DIVISIÓN DE EDUCACIÓN CONTINUA
FACULTAD DE INGENIERÍA U.N.A.M.
CURSOS ABIERTOS

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA, UNAM
CURSOS ABIERTOS

CURSO: **CC065 Sistema Operativo Unix (Parte II) Administración del Sistema**
FECHA: **22 de septiembre al 1º de octubre de 1997**

EVALUACIÓN DEL PERSONAL DOCENTE

(ESCALA DE EVALUACIÓN 1 A 10)

CONFERENCISTA	DOMINIO DEL TEMA	USO DE AYUDAS AUDIOVISUALES	COMUNICACIÓN CON EL ASISTENTE	PUNTUALIDAD
ING. JESSICA BRISEÑO CORTES				

Promedio _____

EVALUACIÓN DE LA ENSEÑANZA

CONCEPTO	CALIF
ORGANIZACIÓN Y DESARROLLO DEL CURSO	
GRADO DE PROFUNDIDAD DEL CURSO	
ACTUALIZACIÓN DEL CURSO	
APLICACIÓN PRACTICA DEL CURSO	

Promedio _____

EVALUACIÓN DEL CURSO

CONCEPTO	CALIF
CUMPLIMIENTO DE LOS OBJETIVOS DEL CURSO	
CONTINUIDAD EN LOS TEMAS	
CÁLIDAD DEL MATERIAL DIDÁCTICO UTILIZADO	

Promedio _____

Evaluación total del curso_____

Continúa...2

1. ¿Le agradó su estancia en la División de Educación Continua?

SI ☐ NO ☐

Si indica que "NO" diga porqué: _____

2. Medio a través del cual se enteró del curso:

Periódico *Excélsior*	
Periódico *La Jornada*	
Folleto anual	
Folleto del curso	
Gaceta UNAM	
Revistas técnicas	
Otro medio (Indique cuál)	

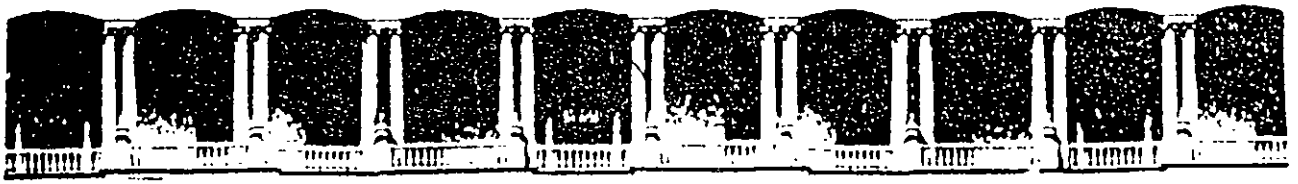
3. ¿Qué cambios sugeriría al curso para mejorarlo?

4. ¿Recomendaría el curso a otra(s) persona(s) ?

SI ☐ NO ☐

5.¿Qué cursos sugiere que imparta la División de Educación Continua?

6. Otras sugerencias:



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

SISTEMA OPERATIVO UNIX (PARTE II)

ADMINISTRACION DEL SISTEMA

22 de septiembre al 1º de octubre de 1997

DIRECTORIO DE PROFESORES

ING. JESSICA BRISEÑO CORTES
ASESOR PARTICULAR
NORTE 91 No. 4729
COL. NUEVA TENOCHTITLAN
DELEGACION GUSTAVO A. MADERO
C.P. 07890 MEXICO,D.F.
TEL: 751 92 56

'pmc.

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

SISTEMA OPERATIVO UNIX (PARTE II)

ADMINISTRACION DEL SISTEMA

MATERIAL DIDACTICO

SEPTIEMBRE - OCTUBRE 1997

Palacio de Mineria Calle de Tacuba 5 Primer piso Deleg. Cuauhtemoc 06000 Mexico, D.F. APDO. Postal M 2285
Telefonos: 512-8955 512-5121 521-7335 521-1987 Fax 510-0573 521-4020 AL 25

Capítulo 1

ACTIVIDADES DEL ADMINISTRADOR DEL SISTEMA

1. El administrador del Sistema

El administrador del sistema es la persona responsable de la operación de un sistema UNIX. El administrador debe ser una persona (algunas veces se trata de un conjunto de personas) que tenga conocimiento amplio de UNIX, así como de las tareas que debe realizar como tal. Algunas de las tareas que debe llevar a cabo se mencionan a continuación, algunas de ellas las puede llevar a cabo alguna otra persona con autorización del administrador.

Instalación y configuración del sistema

- Análisis de requerimientos del sistema
- Instalación del sistema operativo
- Instalación y reconfiguración de Hardware
- Instalación y configuración del software
- Verificación del sistema

Actividades periódicas

- Respaldos.
- Administración de usuarios del sistema.
- Administración de los dispositivos periféricos del sistema (impresoras, unidades de cinta, terminales, etc.)
- Administración de los recursos de la computadora (disco, memoria, etc.)
- Administración de procesos
- Apagar y encender la máquina

Actividades eventuales

- Instalación y configuración de software
- Instalación y configuración de hardware (impresoras, terminales, tarjetas, discos, etc.)
- Modificación del performance del sistema
- Instalación de nuevas versiones del sistema operativo
- Mantenimiento del sistema de archivos
- Reconfiguración del kernel
- Configuración de las conexiones a redes
- Implantación de esquemas de seguridad
- Automatización de tareas y procedimientos de administración
- Resolución de problemas del sistema

2. Privilegios de Superusuario

UNIX proporciona un esquema de sistema de archivos y procesos, en donde cada usuario es dueño de sus archivos y procesos, de tal forma que otros usuarios no pueden interferir si estos no tienen permisos; sin embargo, el administrador del sistema tiene la necesidad de violar estos permisos en algunas situaciones:

- Procesos inactivos y con mucho tiempo de CPU
- Sesiones "fantasmas"
- Bloqueo de procesos
- Archivos "basura" de usuarios
- etc.

Para cumplir con estas necesidades y con todas las tareas que como administrador se tienen, debe de contar con los privilegios de un usuario especial identificado con el UID 0, generalmente es **root.**

El usuario root puede llevar a cabo cualquier acción sobre archivos y procesos, algunos comandos y procedimientos solo los puede realizar root:

- Montar y desmontar Sistemas de Archivos
- Matar procesos
- Crear archivos de dispositivos
- Cambiar la fecha del sistema
- Establecer limites en el uso de recursos
- Shutdown
- Funciones como: reconfiguración del kernel, backups, reconfiguración de archivos de inicialización, etc.

3. El usuario root

El usuario root es un usuario especial en UNIX y las tareas y comandos que se ejecuten por él pueden resultar criticas para el sistema, es por ello que se deben establecer algunas normas para evitar que cualquier personas se pueda identificar como root.

Al igual que otros usuarios el acceso al sistema como root esta controlado por una contraseña, algunas consideraciones importantes que se pueden tomar son las siguientes:

- Procurar que solamente el administrador del sistema o un conjunto reducido de personas conozcan la contraseña de root
- Escoger una contraseña que no sea fácil de adivinar
- Cambiar la contraseña periódicamente

Otra forma de adquirir los privilegios de root es identificándose como tal por medio del comando **su** (set user identity). Cuando se proporciona este comando sin argumentos se crea un proceso de shell con privilegios de root (siempre y cuando se haya proporcionado la contraseña adecuada).

Ejemplo:

```
% su
password:
#
```

4. Reglas de seguridad para el usuario root

Para evitar que cualquier persona pueda identificarse como root y para monitorear las tareas que se llevan a cabo como este usuario se pueden tomar las siguientes medidas de seguridad:

- Establecer una buena contraseña
- Cambiar periódicamente la contraseña de root
- Especificar las terminales desde las cuales alguien se puede identificar como root (generalmente solo la consola)
- Restringir el uso del comando su:

  chmod 500 /bin/su

- En caso de permitir el uso del comando su a algunos usuarios, revisar periódicamente los archivos /usr/adm/sulog y /usr/adm/OLDsulog para conocer quienes ejecutaron el comando su y si tuvieron éxito o no.
- Revisar periódicamente el archivo /etc/btmp para conocer los intentos fallidos de conexión al sistema.
- Obligar a que como root se utilicen rutas absolutas para la ejecución de ciertos comandos, esto no colocando en el path de root ".".
- Cuando hay conexión en una red con TCP/IP evitar el archivo .rhosts en el directorio root (/).

5. Otros usuarios importantes

Existentes otros usuarios en el sistema que tienen ciertos privilegios y que son utilizados para ciertos propósitos especiales.

bin	El usuario bin es el dueño de los directorios que contienen los comandos del sistema.
sys	En algunos sistemas, el usuario sys es dueño de algunos archivos especiales.
kmem	En algunos sistemas se utiliza a el usuario kmem en lugar de sys.
daemon	El usuario daemon es utilizado para tomarse como dueño de algunos comandos que necesitan acceso restringido a archivos.

Todos estos usuarios especiales son registrados en el archivo /etc/passwd y es conveniente que se coloque en el campo asignado al password un carácter especial, por ejemplo "*", para evitar que alguien pueda identificarse como alguno de ellos.

Capítulo 2

BOOT Y SHUTDOWN

1. Proceso de Boot

El proceso de Boot o Bootstrapping es el que permite inicializar la computadora para que el sistema operativo pueda administrar sus recursos.

Es importante que el Administrador conozca los pasos que se llevan a cabo en este proceso, ya que en éste se muestra información de la configuración de la máquina, se carga el sistema operativo a memoria y se revisan los Sistemas de Archivos, además de algunas otras tareas que afectan directamente la operación del sistema.

Muchos sistemas UNIX proporcionan un mecanismo residente que permite controlar, al momento de encender la máquina, el proceso de boot. En computadoras personales aparece un indicador como el siguiente:

boot:

el cual permite inicializar el boot de disco o diskette al proporcionar ENTER o bien cargar MS-DOS al teclear "dos".

En otros sistemas como minicomputadoras y estaciones de trabajo, aparece un indicador (generalmente >) desde el cual se pueden proporcionar una serie de comandos que permiten configurar la máquina para que haga el boot desde disco, cinta, o algún otro dispositivo, además de controlar el que se haga el procedimiento en forma automática o no, ejemplo:

> boot

Los pasos que se llevan a cabo durante el proceso de boot se pueden dividir en los siguientes:

- Carga de UNIX en memoria
- Inicialización del kernel
- Pruebas de configuración de hardware
- Creación de procesos
- Operación en modo mono-usuario
- Ejecución de archivos de inicialización
- Operación multi-usuario

Carga de UNIX en memoria

UNIX es un programa, el proceso de boot (coordinado por un programa residente), antes que nada carga UNIX a memoria.

El programa que se carga se conoce como kernel y su nombre depende del UNIX del que se trate, generalmente es /unix o /vmunix.

Algunos sistemas UNIX implementan esta primera tarea en dos pasos. Primero se carga un pequeño programa llamado /boot y se ejecuta. Este programa es el encargado de la carga del kernel a memoria.

En esta etapa no puede existir intervención alguna por parte del administrador.

Inicialización del kernel

En esta etapa, el kernel comienza su ejecución e inicialización.

El kernel revisa cuanta memoria existe disponible, para determinar si es la suficiente para poder residir en memoria.

Si existe memoria suficiente para la ejecución del kernel, se despliega un mensaje reportando la cantidad de memoria utilizada para cargar el kernel y la restante una vez que el kernel se ha cargado.

Pruebas de configuración del hardware

En esta etapa, el kernel revisa cuales son los dispositivos conectados a la máquina y muestra un reporte de ellos.

Los dispositivos que se encuentran registrados pero por alguna causa no responden a estas pruebas de inicialización, son marcados como desabilitados.

La información de dispositivos de hardware que se reportan son normalmente:

- En caso de computadoras personales, número de puertos seriales, de puertos paralelos características de cada uno de ellos.

- Discos conectados, capacidad de cada uno de ellos, tipo, controladores, etc.

- Memoria e información de Sistemas de Archivos:

  Memoria reservada para el kernel
  Memoria disponible
  Dispositivo del Sistema de Archivos root

- Interfases de red.

- Información de unidades de almacenamiento secundario:

  Tipo
  Nombre del dispositivo
  Características

Creación de procesos

En esta etapa, el kernel crea algunos procesos "daemons" que son útiles durante el proceso de boot. El número y la naturaleza de estos procesos depende del sistema UNIX del que se trate. En sistemas BSD los procesos que se crean son los siguientes:

- swapper (pid 0): controla los page faults que se dan cuando se llevan a cabo referencias a memoria por medio de procedimientos de swap.

- init (pid 1): es el proceso más importante en el sistema, es el que controla el nivel de ejecución del sistema operativo y los procesos del mismo.

- pagedaemon (pid 2): controla la localización de páginas en el área de swap y memoria.

Operación en modo mono-usuario

En esta etapa, el proceso de init crea un proceso de Bourne shell y se transfiere el control a la consola como usuario root. Después de que se termina esta sesión se continúa con la siguiente etapa.

En modo mono-usuario se pueden llevar a cabo algunas tareas de administración importantes las cuales no es conveniente realizar en modo multiusuario:

- Revisión de los sistemas de archivos con **fsck**
- Montar sistemas de archivos
- Establecer la fecha del sistema

En modo mono-usuario:

- No existen procesos en ejecución
- No existen sesiones de usuarios
- Solamente el Sistema de Archivos de root se encuentra montado

Ejecución de archivos de inicialización

El siguiente paso en el proceso de boot es la ejecución de ciertos archivos por medio de los cuales se configura el sistema, el nombre de estos archivos depende del sistema UNIX del que se trate, pero en forma general se puede decir que para sistemas BSD los archivos son el /etc/rc y /etc/rc.local, mientras que en sistemas UNIX de SYSTEM V existe el /etc/rc y /etc/rc[012] dependiendo del nivel que se quiera ejecutar.

Alguna de las tareas que se llevan a cabo en estos archivos son las siguientes:

- Establecer el nombre de la computadora
- Establecer la fecha del sistema
- Revisión de los sistemas de archivos con fsck
- Habilitación de particiones
- Limpieza del directorio /tmp
- Inicialización de procesos "daemon" para red
- Inicialización del sistema de contabilidad y quotas
- Configuración de las interfases de red
- Información de áreas de swap adicionales
- Otras actividades

2. Proceso de Shutdown

El proceso de shutdown es la forma en que se da de baja el sistema. Existen varios métodos de llevar a cabo esto:

- Apagar la máquina
- con el comando shutdown
- con el comando halt
- con el comando reboot
- con el comando init
- terminando el proceso init

Apagando físicamente la máquina.

Este no es un método recomendable, ya que el apagar la máquina sin llevar a cabo algún procedimiento que dé de baja al sistema, puede provocar que los sistemas de archivos queden en un sistema inconsistente y que incluso, se llegue a dañar algún disco.

El comando Shutdown

El comando shutdown es la forma más conveniente de dar de baja el sistema. La forma de invocarlo, en sistemas BSD es la siguiente:

```
% shutdown tiempo [mensaje]
```

En sistemas de SYSTEM V se invoca:

```
% shutdown [-y] [-gtiempo] [su]
```

En ambos casos el tiempo se expresa en minutos, o bien, se puede utilizar la palabra "now" para llevar a cabo el comando inmediatamente.

Se pueden utilizar opcionalmente -r para indicar un reboot o -h para dar de baja completamente el sistema.

El comando halt

Este comando permite dar de baja completamente el sistema de una forma mucho más rápida que shutdown.

Es conveniente utilizarlo solamente en casos de emergencia y fallas.

El comando reboot

Este comando reinicializa el sistema después de dar un halt.

El comando init

El comando init permite un cambio en el nivel de operación en el sistema operativo, la forma de indicarlo es la siguiente:

init [nivel]

donde el nivel es alguno de los siguientes:

- 0: shutdown
- 1: modo mono-usuario
- 2: modo multi-usuario
- S,s: modo single-user

ADMINISTRACIÓN DE USUARIOS

1. Sistemas de administración de usuarios

Muchos sistemas incluyen programas que permiten crear una cuenta de usuarios; algunos otros incluyen programas que automatizan las tareas del administrador, entre ellas la de crear cuentas de usuario. Algunos ejemplos son:

- sysadmsh (SCO UNIX System V).
- sam (HP/UX).
- smith (AIX).
- sysadmin (EP/IX).
- adduser (Ultrix).

2. Alta de una cuenta de usuario

Las tareas que se llevan a cabo para generar una cuenta de usuario son:

- Editar el /etc/passwd.
- Crear un directorio de HOME.
- Editar el /etc/group.
- Copiar archivos de configuración.
- Asignar un password.
- Asignar recursos.

Edición del /etc/passwd

Archivo de registro de los usuarios validos en el sistema. La información que contiene es la siguiente:

1 Nombre de la cuenta de usuario.

2 Password encriptado.

3 Identificador de Usuario (UID).

4 Identificador del Grupo de default (GID).

5 Información particular del usuario.

6 Directorio de HOME.

7 Shell de inicio.

Ejemplos de registros del /etc/passwd:

```
acf:g/GQFQc2mugIs:280:15:J. Antonio Chaves FI:/usr/cecafi/acf:/bin/csh
aaron:gfL..Qc2mugxw:315:15:Aaron Arcos Tapia, CECAFI:/usr/cecafi/aaron:/bin/csh
jess:aN1/3jkImuped:516:15:C. Jessica Brisenio C., CECAFI:/usr/cecafi/jess:/bin/sh
nam:wpl109smn sls:785:15:Norberto Arrieta M., CECAFI:/usr/cecafi/nam:/bin/csh
```

- **Nombre de la cuenta de usuario**

  - Debe ser única.

  - Debe contener solamente letras y numeros.

  - Su longitud no debe ser mayor a 8 caracteres.

  - El administrador debe elegir un esquema de asignación de nombres.

- **Password encriptado**

  - El password para una clave se asigna con el comando passwd.

  - Cuando se edita el /etc/passwd para dar de alta una clave, se debe colocar "X" en el campo de passwd.

  - Si el campo esta vacío indica que la clave no tiene password.

- **Identificador de Usuario (UID)**

  - Debe ser único.

  - Para SCO UNIX System V el valor esta entre 0 y 32767, para HP/UX entre 0 y 60000.

  - En ambiente de red el UID para claves iguales en diferentes nodos debe ser el mismo.

  - No es necesario que el /etc/passwd este ordenado por el UID.

  - La clave de root tiene el UID 0.

- **Identificador del Grupo de default (GID)**

  - Para SCO UNIX System V el valor esta entre 0 y 32767, para HP/UX entre 0 y 60000.

  - El GID 0 se reserva para el grupo de root.

- **Información particular del usuario**

  - La información puede incluir: nombre del usuario, dirección, teléfono, nombre de su empresa, etc.

  - Puede contener espacios en blanco.

- **Directorio de HOME**

  - Debe de contener toda la ruta del directorio.

- **Shell de inicio**

  - Indica el shell que se ejecutara cuando el usuario entre a sesión.

  - Puede ser /bin/csh, /bin/sh, /bin/ksh o cualquier otro programa, necesariamente un interprete de comandos.

Creación del directorio de HOME

- Por convención el nombre del directorio de home es el mismo que el nombre de la cuenta de usuario.

- El directorio padre de los directorios HOME de las claves de usuario varia de sistema a sistema, generalmente es /usr o /users.

- Para crear el directorio de HOME para la clave de usuario curso100:

```
mkdir /usr/users/curso100
chown curso100 /usr/users/curso100
chgrp users /usr/users/curso100
```

Edición del /etc/group

- Cada registro del archivo /etc/group contiene:

  - Nombre del grupo
  - Password (este campo ya no es utilizado)
  - Identificador del grupo (GID)
  - Lista de claves de usuario pertenecientes al grupo (separadas por comas).

- El nombre de cada grupo esta formado por 8 caracteres a lo mas.

- No es necesario que las claves de usuario se indiquen como pertenecientes a su grupo de default en el /etc/group. Solamente se indica cuando una cuenta pertenece a grupos adicionales.

- Ejemplos de registros del /etc/group:

```
users::50:
cecafi::15:acf,nam,jess,aaron
```

Copia de archivos de configuración

Cuando un usuario entra a sesión es recomendable que configure su medio ambiente, para ello existen una serie de archivos que llevan a cabo esta tarea. Además se cuenta con otros archivos que permiten configurar una sesión del editor vi o bien una de mail. Algunos de estos archivos son:

.cshrc	permite configurar un ambiente de csh.
.login	permite configurar el medio ambiente de inicio de sesión cuando el shell que atiende es csh.
.profile	tiene la misma función que .login pero cuando se trata de bourne shell o kshell.
.logout	permite ejecutar acciones cuando se termina la sesión, siempre y cuando el interprete de comandos sea csh.
.mailrc	configura una sesión de mail.
.exrc	configura una sesión de vi.
.hushlogin	es un archivo vació que impide que a un usuario le aparezcan los mensajes desplegados por /etc/motd.

Existen prototipos de archivos de configuración realizados para que únicamente se copien al directorio de HOME de las cuentas de usuario. El lugar en donde se localizan depende del sistema:

- En ultrix se localizan en el directorio /usr/skel.

- En HP/UX se localizan en /etc y son llamados d.login, d.profile, etc.

- En SCO UNIX System V en el directorio /usr/lib/mkuser existen directorios para cada tipo de shell conteniendo sus archivos de configuración.

Es importante que estos archivos se modifiquen para que configuren el medio ambiente de usuario adecuadamente, es conveniente:

- Incluir en el PATH los directorios de utilerías y comandos importantes.

- Definir variables como:

  - TERM
  - history
  - prompt

- Definir alias en los comandos rm y mv para evitar borrados indeseables:

```
alias rm "rm -i"
alias mv "mv -i"
```

Ejemplos de estos archivos de configuración se muestran a continuación:

.login

```
setenv SHELL /bin/csh

set ignoreeof                                   # don't let control-d logout
set path = ($path $home/bin .)      # execution search path

set noglob
set term = (`tset -m ansi:ansi -m :\?ansi -r -S -Q`)
if ( $status == 0 ) then
        setenv TERM "$term"
endif
unset term noglob

/usr/bin/prwarn                 # issue a warning if password due to expire
```

.cshrc

```
set noclobber                                   # don't allow '>' to overwrite
set history=20                                  # save last 20 commands
if ($?prompt) then
        set prompt=\!%\                 # set prompt string
# some BSD lookalikes that maintain a directory stack
        if (! $?_d) set _d = ()
        alias   popd   'cd $_d[1]; echo ${_d[1]}:; shift _d'
        alias   pushd          'set _d = (`pwd` $_d); cd \!*'
        alias   swapd          'set _d = ($_d[2] $_d[1] $_d[3-])'
        alias   flipd    'pushd .; swapd ; popd'
endif
alias print 'pr -n \!:* | lp'                   # print command alias
```

.profile

```
PATH=$PATH:$HOME/bin:.                    # set command search path
MAIL=/usr/spool/mail/`logname`            # mailbox location
export PATH MAIL

# use default system file creation mask

eval `tset -m ansi:ansi -m :\?${TERM:-ansi} -r -s -Q`

/usr/bin/prwarn                           # issue a warning if password due to expire
```

.mailrc

```
unset askcc
set ask
set autoprint
```

.exrc

```
set autoindent
set nomesg
set number
set scroll=24
set shell=/bin/csh
```

Asignar un password

- Para asignar un password se utiliza el comando passwd.

- El password asignado debe de ser una palabra que no sea fácil de descifrar; pero que sea fácil de recordar por el usuario.

- Algunos sistemas obligan a que el password tenga ciertas características, como longitud, caracteres que lo deben formar, etc.

- En SCO UNIX System V:

  - El archivo /etc/default/passwd establece una serie de parámetros para el manejo de passwords.

  - Se establece un tiempo mínimo para poder cambiar un password, un tiempo máximo para cambiar el password.

  - Si el password no se cambia a tiempo se bloquea la cuenta.

  - Se establece un parámetro para indicar el numero máximo de login's fallidos, después de lo cual se bloquea la clave.

  - Se tiene la opción de generar el password automáticamente.

3. Administración de usuarios

Algunas consideraciones importantes que se pueden llevar a cabo para la administración de los usuarios son las siguientes:

- Evitar el que una cuenta de usuario no tenga password. Para saber que cuentas estan en este estado:

  ```
  grep '[^:]*::' /etc/passwd
  ```

- Editar el archivo /etc/motd para indicar mensajes importantes a los usuarios; siempre y cuando no cuenten con el archivo .hushlogin en su directorio HOME.

- Revisar la actividad de los usuarios con ayuda del comando **finger**, para de esta forma terminar sesiones que han permanecido inactivas durante un largo período de tiempo.

- Desactivar cuentas de usuario, reemplazando el password en /etc/passwd con un carácter especial, por ejemplo "*".

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

SISTEMA OPERATIVO UNIX (PARTE II)

ADMINISTRACION DEL SISTEMA

MATERIAL DIDACTICO

SEPTIEMBRE - OCTUBRE 1997

Palacio de Mineria Calle de Tacuba 5 Primer piso Deleg Cuauhtemoc 06000 México D.F. APDO Postal M-2285
Teléfonos 512-8955 512-5121 521-7335 521-1987 Fax 510-0573 521-4020 AL 22

INTRODUCCION

Todos los propveedores de hardware proporcionan un **spooling system** que sigue las convenciones del ATT Spooling System, el BSD Spooling System o es una combinación de ambos.

Para determinar el tipo de spooling system en el que se esta trabajando, basta con verificar cuál es el planificador de trabajos de impresión: lpd para BSD o lpsched para ATT, ya que muchos sistemas proporcionan interfaces para ambos tipos de spooling system.

TERMINOS IMPORTANTES

Nombre de Impresora

Para configurar una impresora en el **Line Printer Spooling System** es necesario identificarla por un nombre. El nombre puede contener hasta 14 caracteres alfanuméricos (incluyendo '_').

Clase de impresora

Es un conjunto de impresoras que son tratadas como un grupo.

Destino de impresión

El destino de impresión es el nombre de una impresora o el nombre de una clase de impresoras. Cuando se especifica una clase como destino de impresión, los trabajos de impresión pueden ser enviados a cualquier impresora que pertenezca a la clase.

Impresora de default

Es el destino de impresión al cual se envían los trabajos para los cuales no se especifica un destino. Puede o no existir una impresora de default.

Cada usuario puede configurar su ambiente de trabajo para utilizar una impresora por default. Para ello debe crear la variable de ambiente LPDEST en ambientes ATT o la variable PRINTER para ambientes BSD:

En csh:

```
setenv LPDEST nombre
```

En sh:

```
set LPDEST=nombre
export LPDEST
```

Archivo de dispositivo

Los archivos de dispositivos no son parte del Line Printer Spooling System. Son archivos especiales que definen la comunicación con un dispositivo físico. Los archivos de dispositivo para las impresoras son: /dev/lp, /dev/lp0, /dev/lp1, etc.

BSD SPOOLING SYSTEM

El Spooling System en sistemas BSD es un tanto primitivo, soporta solamente operaciones simples para el manejo de impresoras, es díficil su mantenimiento y muchas veces no funciona adecuadamente.

La gran ventaja de este sistema de impresión es que el modelo esta diseñado para poder adecuarse fácilmente a una red heterogénea, permitiendo el que una impresora pueda ser utilizada por varios nodos.

En este sistema de impresión, todos los trabajos de impresión son controlados por el proceso daemon **lpd** (generalmente localizado en /usr/lib) y el programa **lpr.**

lpr accepta trabajos de impresiónn y los coloca en un directorio de spool, donde el procesoo lpd busca para imprimir los trabajos. lpr es el único programa que puede encolar trabajos de impresión, otros programas como **pr** o **print,** simplemente llaman a lpr.

Cuando el programa lpr es ejecutado, se determina la impresora destino del trabajo de impresión. La impresora destino puede ser especificada con la opción -P del programa lpr:

lpr -Pprinter_name

Si no se especifica la impresora destino, el nombre de esta se toma de la variable de ambiente **PRINTER**, si la variable no ha sido creada, entonces el trabajo de impresión se envía a la impresora configurada como default.

Una vez que el comando lpr conoce el nombre de la impresora destino, este la busca en el archivo /etc/printcap. Este archivo indica, entre otra información, el directorio donde los trabajos de impresión, dirigidos a la impresora referenciada, deben ser colocados. Este "directorio de spool" es generalmente **/var/spool/**printername.

lpr crea dos archivos en el directorio de spool para cada trabajo de impresión:

- **Archivo de control**, identificado como cf (control file) seguido por un número que identifica al trabajo de impresión. Este archivo contiene referencias e información del trabajo de impresión, como por ejemplo, el identificador del usuario que envia el trabajo.

- **Archivo de datos**, identificado por df (data file) seguido por el mismo número de identificación que el archivo de control. Este archivo contiene los datos a ser impresos.

El identificador númerico asignado a los anteriores archivos puede ser de solamente tres dígitos, por lo que se pueden tener problemas cuando se tiene un número mayor a 999 trabajos de impresión.

Después de que se han creado los archivos en el directorio de spool, el programa lpr notifica a lpd de la existencia del trabajo.

Cuando lpd recive la notificación, este consulta el archivo /etc/printcap para determinar si la impresotra destino es una impresora local o una remota. Si se trata de una impresora local, lpd crea una copia de si mismo para atender al trabajo; si la impresora es remota, se establece una conexión al nodo remoto y se transfiere el archivo de control y el archivo de datos y estos son borrados localmente.

Los trabajos de impresión son atendidos de forma FIFO, pero el administrador del sistema puede modificar este esquema con el comando **lpc.**

Cuando el trabajo esta listo para ser impreso, lpd crea una serie de **pipes** hacia el dispositivo de impresión en los cuales lleva a cabo un proceso de filtrado de los datos.

El proceso de filtrado puede llevar a cabo formateo de datos para soportar protocolos particulares a un tipo de impresora.

El archivo /etc/printcap

El archivo /etc/printcap contiene información que describe a las impresoras a donde se pueden mandar trabajos de impresión.

Para cada impresora se indica en primera instancia, una lista de nombres mediante los cuales se puede identificar a la impresora, la lista es separada por el caracter "|". Después de la lista de identificadores aparece información referente a:

- Especificaciones de archivos y directorios de spool
- Información para impresión remota
- Filtros
- Opciones para comunicación
- Información de paginación

lpq

El comando lpq permite examinar los trabajos de impresión que se encuentran encolados. Normalmente es usado con la opción -P para obtener la información de la cola de impresión para una impresora en particular. Se utiliza la impresora por default cuando se omite la opción -P:

```
% lpq
laser is ready and printing
Rank  Owner      Job  Files      Total Size
active acf       314  sort.c     29875 bytes
1st   jess       286  bd.001      1769 bytes
2st   jess        12  bd.002     54616 bytes
%
```

La primera columna indica el orden de impresión. Si el primer trabajo no esta activo, entonces el daemon de impresión no esta en ejecución.

La segunda columna indica el dueño del trabajo.

La tercera columna indica el número de trabajo. Este número es importante para hacer referencia al trabajo cuando se quiera borrar o modificar su estado en la cola.

La cuarta columna indica el o los archivos a ser impresos. Cuando la salida a ser impresa provien de un pipe, aparece el identificador *standar input.*

La última columna indica el tamaño del trabajo de impresión antes de ser filtrado.

lprm

El comando lprm permite borrar un trabajo de la cola de impresión:

lprm *jobid* borra el trabajo con identificador igual a jobid

lprm *user* borra los trabajos de impresión de un usuario

lprm borra el trabajo activo

lprm - borra los trabajos enviados por el usuario que ejecuta el comando, en el caso de roor se borran todos los trabajos de impresión.

lpc

El comando lpc permite llevar a cabo las siguientes tareas:

- Habilitar o desabilitar la cola de impresión para una impresora en particular
- Habilitar o desabilitar la impresión para una impresora en particular
- Borrar trabajos de una cola de impresión
- Mover un trabajo de impresión a la primera posición
- Manipular el lpd daemon
- Obtener información de status de una impresora

ATT SPOOLING SYSTEM

El ATT Spooling System es originalmente más robusto y más utilizado que el de BSD, desafortunadamente no fue diseñado para adaptarse a un esquema de impresión remota.

Para imprimir en este sistema de impresión se debe invocar directa o indirectamente el comando **lp.** lp toma los datos de impresión y los coloca en el directorio de spool apropiado. El daemon lpsched determina cuando y donde debe ser impreso un trabajo, para ejecutar un archivo de interface que formatea los datos para una impresora en particular.

Archivos de interfaces

Cada impresora que ha sido configurada en el sistema tiene asociado un archivo de interface, el cual es utilizado como un filtro para los trabajos que se le envían. Dicho archivo es un script en shell el cual puede ser modificado por el administrador para adecuarlo a las necesidades de la impresora.

Los archivos de interfaces se encuentran en el directorio /usr/spool/lp/interface.

Existen archivos de interface "modelo" para los tipos de impresoras mas usuales, estos archivos se encuentran en el directorio /usr/spool/lp/model.

lp y lpsched

lp es el comando comúnmente utilizado para mandar trabajos a impresión. Este comando toma los trabajos de impresión y los coloca en un spool de impresión. Cada uno de los trabajos de impresión se colocan en un archivo en el directorio /usr/spool/impresora. Dichos archivos tienen el nombre impresora-N donde N es un numero de identificación asignado por lp.

lpsched (line printer scheduler) es el proceso daemon que toma los archivos colocados en el directorio /usr/spool y los envía al dispositivo apropiado, controlando el flujo de datos al dispositivo físico.

Para arrancar lpsched:

/usr/lib/lpsched

para parar el daemon:

/usr/lib/lpshut

El archivo /usr/spool/lp/SCHEDLOCK es creado al momento de arrancar lpsched para asegurar que solamente existe una copia de este proceso en ejecución. Cuando se para el daemon el archivo es borrado.

Configuración de impresora

1. Asegurarse de tener privilegios de root.

2. Asegurarse de que no exista el proceso de lpsched:

   /usr/bin/lpsched -r

3. Utilizar el comando lpadmin para añadir la impresora al Line Printer Spooling System.

4. Habilitar una cola de impresión para la impresora.

   /usr/lib/accept impresora

5. Habilitar la impresora:

   /usr/lib/enable impresora

6. Arrancar lpsched.

7. Si se desea establecer una impresora de default:

   /usr/lib/lpadmin -dimpresora

lpadmin

Para dar de alta una impresora en el Line Printer Spooling System se utiliza el comando /usr/lib/lpadmin con la siguiente sintaxis:

/usr/lib/lpadmin -p*impresora* -v*dispositivo* { -e*pr* | -m*mod* -i*int* } [-c*class*] [{-l | -h}]

donde:

- *impresora* es el nombre de la impresora.

- *dispositivo* es el dispositivo asociado.

- Se utilizan las opciones e, m o i para especificar el archivo de interface a ser utilizado con la impresora:

  e especifica un archivo de interface ya existente para una impresora.

  m toma el archivo del directorio /usr/spool/lp/model.

  i especifica un archivo en especifico.

- *class* es la clase a la que pertenece la impresora. Si no existe la clase, se crea automáticamente.

- las opciones l y h indican si la impresora esta conectada como terminal o si esta conectada por un puerto dedicado para la impresora.

Baja de impresora

1. Asegurarse de tener privilegios de root.

2. Desabilitar la impresora:

   /usr/lib/disable impresora

3. Cancelar la cola de impresión:

   - Impedir que se acepten trabajos de impresión:

     /usr/lib/reject impresora

   - Mover los trabajos de impresión a otra cola:

     /usr/lib/lpmove impresora impresora2

   - Abortar los trabajos en la cola de impresión:

     /usr/lib/cancel impresora

4. Dar de baja lpsched:

   /usr/lib/lpshut

5. Dar de baja la impresora del Line Printer Spooling System:

   /usr/lib/lpadmin -ximpresora

6. Arrancar lpsched.

Monitoreo de trabajos de impresion

El comando lpstat es utilizado para mostrar diferentes aspectos relacionados con los trabajos de impresión.

El comando lpstat sin argumentos muestra los trabajos de impresión para todas las impresoras. Además se tienen las siguientes opciones para dicho comando:

-r muestra el estado de lpsched.

-d muestra la impresora de default.

-cclass muestra las impresoras pertenecientes a una clase.

-uuser muestra el estado de los trabajos de impresión de un usuario.

-pimp muestra el estado de una impresora.

-vimp muestra el dispositivo asociado con una impresora.

-adest muestra si un destino de impresión acepta trabajos.

-s muestra las impresoras instaladas, la clase a la que pertenecen así como el dispositivo al que están conectadas.

-t muestra la información presentada por la opción s además del estado de cada destino de impresión.

ADMINISTRACION DE REDES TCP/IP

1. Instalación física

- Crear y configurar la interface de red de manera que ésta sea reconocida por el sistema operativo.

- Adecuar el ambiente para permitir el correcto funcionamiento de los servicios de red (mail, ftp, telnet, etc.)

- Monitoreo y aspectos de seguridad

Aspectos a considerar:

- El modelo de la interface de red que se está utilizando

- La dirección IP que se le piensa asignar al nodo y su tipo
(conocer dirección de la red)

- El nombre que se le piensa asignar al nodo incluyendo el dominio al que pertenece

- Los posibles seudónimos para el nombre del nodo

- La máscara de subred que se utiliza en el sistema autónomo

- La dirección de broadcast

- Los nodos confiables en la red

El modelo de la interface de red determina el tipo de hardware que se utiliza para conectar el nodo a la red. Para lograr un comportamiento óptimo del sistema se requiere reconfigurar el kernel con los nuevos datos de la interface, así como modificar algunos parámetros.

Si no hay conexión a la Internet se sugiere que la dirección de red sea 130.1 (de clase B) y de nodo 0.1, es decir 130.1.0.1.

La dirección escogida y el nombre del nodo hay que incluirlos en el archivo /etc/hosts.

Si no hay conexión a la red se recomienda utilizar la dirección de subred 255.255.0.0 o, como lo solicitan algunos procedimientos de instalación, los últimos 16 bits. Hay que saber también si la dirección de broadcast está formada únicamente por unos o por ceros. En la versión 4.2 de BSD son ceros, pero a partir de la versión 4.3, y en forma general, son unos. En ocasiones se solicita también el nombre de la red para utilizarlo como mnemónico en algunos comandos como ifconfig, ésto es equivalente a editar el archivo /etc/networks.

Una vez hecho ésto, hay que proceder a cambiar el archivo de inicialización de la red, /etc/rc.local. Para dar de alta la interface con los parámetro arriba mencionados, se utiliza el comando ifconfig:

% /etc/ifconfig interface dirección dirección-destino parámetros

Como se mencionó, este comando asocia una interface con una dirección IP y permite la configuración de los parámetros que afectan a la interface. Entre éstos parámetros tenemos la dirección de broadcast, la máscara de red utilizada, el tipo de protocolo de resolución de direcciones y otros. Las líneas de comandos que se utilizan comúnmente son de la forma:

```
/bin/hostname atl.cecafi.unam.mx
/etc/ifconfig ln0 `/bin/hostname` broadcast 132.248.54.255 \
netmask 255.255.0.0 arp up
/etc/ifconfig lo0 localhost
```

La primera línea de comandos define el nombre del nodo. A continuación el comando ifconfig configura la interface conectada a la red (se observa que se especifica como un dispositivo). Es válido especificar el nombre del host (`/bin/hostname` hace precisamente esto) o también dar la dirección en forma puntual (en este ejemplo 132.248.54.2). En el primer caso, el nombre se debe encontrar en el archivo /etc/hosts. La dirección de netmask se especifica bien en forma puntual, o bien utilizando algunos de los nombres especificados en el archivo /etc/networks creado con anterioridad. La opción *arp* indica que la interface utilizará el protocolo de resolución de direcciones físicas ARP (aunque en general no se especifica ya que es el default). Por último, se configura la interface lo0, la de *loopback*, es decir aquella a través de la cual se hacen las comunicaciones internas sin salir al medio. Esta dirección es la 127.0.0.1, que también se pude identificar a través del nombre localhost que se encuentra también en el archivo /etc/hosts. La opción *up*, también default, es para arrancar la interface. El comando:

```
ifconfig <interface>
```

proporciona el valor de los parámetros actuales de la interface. Estos se pueden cambiar si así se desea, pero en general sólo se inicializan cuando arranca la máquina.

2. Protocolos de ruteo

Existen varios protocolos de ruteo:

- EGP External Gayeway Protocol
- RIP Routing Information Protocol (InternalGP)
- IGRP Internal Gateway Routing Protocol (IGP)
- Hello Protocolo de intercambio entre gateways usado originalmente en NSFNet (IGP)

Los protocolos de ruteo utilizan un costo métrico para asignar pesos a cada ruta.

Para llevar a cabo un ruteo estático, se hace uso del comando /etc/route:

```
/etc/routed add `/bin/hostname` localhost 0 #para loopback
/etc/routed add pemex gw-pemex 1  #para alcanzar red pemex
/etc/routed add default gw-default 1  #para cualquier otro
                                      #destino
```

La sintaxis precisa del comando /etc/route se da a continuación:

```
/etc/route [-f] [add, delete] destino gateway númeroSaltos
```

El destino puede ser el nombre de un nodo, el de una red o la palabra clave "default". El gateway es el nombre (dirección) de la máquina a la cual se enviarán los paquetes. El parámetro *númeroSaltos* indica el número de saltos para alcanzar el destino. El argumento -f limpia la tabla de ruteo.

Es muy recomendable incluir todos los comandos necesarios para el ruteo en el archivo /etc/rc.local.

2.1 Ruteo dinámico

Este tipo de ruteo se utiliza en sistemas autónomos con varios gateways, para evitar colocar a mano las rutas de los mismos, en vez de ésto, se utiliza el comando /etc/routed. La siguiente línea se puede añadir al archivo /etc/rc.local si se quiere ruteo dinámico:

```
if [ -f /etc/routed]; then
        /etc/routed & (echo -n ' ruteador') > /dev/console
```

2.2 Ruteo dinámico mejorado

El comando /etc/gated reemplaza a /etc/routed y es capaz de manejar varios protocolos, además de permitir un mapeo de métricas entre protocolos diferentes. Su comportamiento se controla a través del archivo de configuración /etc/gated.conf, el cual permite al administrador tener mucho más control sobre el reconocimiento de rutas, direcciones de broadcast, métricas, y otros aspectos importantes involucrados en estas operaciones. Gated se inicia en el /etc/rc.local de la siguiente forma:

```
if [ -f /etc/routed ]; then
        rm -rf /usr/adm/gatedlog
        /etc/gated -t /usr/adm/gatelog & echo -n ' gated' > /dev/console
```

3. Configuración de las terminales para sesiones remotas

El programa *rlogin* utiliza seudoterminales (software enmascarado como hardware), las cuales deben estar presentes en los archivos de configuración de terminales.

Las seudoterminales, pty, se deben crear en el directorio /dev. Estos son dispositivos de caracteres que utilizan software para emular una terminal. Las características del kernel para soportar este tipo de terminales se incluyen con la línea "pseudo-device pty" en el archivo de configuración. Los nombres deben ser p0-pf para las primeras 16, q0-qf para el segundo grupo y así sucesivamente. Los siguientes comandos crean 32 seudoterminales, cuyos nombre inician con ptyp0 (maestra) y ttyp0 (esclava).

```
cd /dev
MAKEDEV pty0
MAKEDEV pty1
```

3.1 Archivos de configuración de las terminales

Las seudoterminales se deben listar en los archivos de configuración /etc/ttys o /etc/ttytype (el último únicamente si se trata de BSD 4.2). Los elementos típicos de estos archivos se muestran a continuación:

4.3BSD /etc/ttys:

```
ttyp0   none                    network
ttyp1   none                    network
...
ttyqf   none                    network
```

4. Configurando los archivos /etc/hosts.equiv y /.rhosts

La seguridad en una red está controlada por dos archivos: /etc/hosts.equiv, para un control de acceso a nivel de máquina, y ~/.rhosts, para un nivel de seguridad definido por el usuario. El archivo /etc/hosts.equiv especifica las máquinas que son globalmente equivalentes y de confianza. De confianza significa que todos los usuarios, incluyendo a root, en la máquina remota podrán accesar la máquina vía rlogin, rsh, rcp sin especificar un password.

El archivo /etc/hosts.equiv contiene los nombres de las máquinas equivalentes, una por línea: Por ejemplo si atl confiara en kelem y ehecatl, el archivo quedaría como sigue:

```
kelem
ehecatl
```

El archivo ~/.rhosts permite a un usuario en particular especificar las máquinas y los logins en esas máquinas que son equivalentes para él . De manera particular el archivo /.rhosts controla el acceso de root y es la forma más segura de tener una equivalencia para el administrador del sistema.

5. Cargar el Kernel

Una vez que el kernel, los archivos de configuración y los de inicialización hayan sido configurados, se debe cargar el nuevo kernel y probar el software de red.

Desde de ello, no todos los comandos de red trabajarán, pero se pueden probar los mas simples

- ping (/etc/ping) host # verifica la conexión
- telnet host # login en una máquina remota
- ftp host # transferencia de archivos
- rlogin host # login remoto específico de Unix
- rcp # copia de archivos en máquinas remotas
- rsh # ejecución de archivos en máquinas remotas

6. Ligas a rsh

El propósito de crear ligas entre el comando rsh y los nombres de los nodos de la red, es facilitar el uso de algunos comandos. Por ejemplo:

```
cd /usr/hosts
ln /usr/ucb/rsh kelem
```

permite al usuario teclear:

```
kelem
```

en lugar de:

```
rlogin kelem
```

y

```
kelem comando
```

en lugar de:

```
rsh kelem comando
```

El directorio /usr/hosts contiene un script llamado MAKEHOSTS que realiza estas ligas automáticamente del archivo /etc/hosts. Para que los usuarios aprovechen estas abreviaciones, deberán incluir /usr/hosts en su ruta de búsqueda (PATH).

7 El comando rwho

El comando *rwho* es el análogo de who en un ambiente de red. Su funcionamiento de basa en el demonio rwhod, el cual envía periódicamente mensajes a todos los nodos de la red indicando quien ha entrado a sesión. Este demonio recibe también paquetes de otros rwhod's en la red, con los cuales construye una base de datos en /usr/spool/rwho de todos los usuarios y máquinas en la red. Los comandos rwho y ruptime utilizan esta base de datos para proporcionar una fotografía del estado de los usuarios de la red en cada máquina. rwhod se inicializa en el archivo /etc/rc por medio de :

```
if [ -f /etc/rwhod ]; then
        /etc/rwhod;  echo -n ' rwhod' > /dev/console
```

Rwhod puede consumir muchos recursos en una red con tráfico, ya que cada máquina debe escuchar todos los paquetes de broadcast que le llegan. El tiempo de default para realizar los broadcast de rwhod es de 3 minutos, de manera que los datos que reporta rwho y ruptime no son del todo actuales.

8. El demonio inetd

A partir de la versión 4.3 del Unix BSD, las utilerías de red incluyen un servidor maestro que se encarga de controlar los demonios que proporcionan los servicios de red. Este demonio maestro se denomina /etc/inetd y se configura a través del archivo /etc/inetd.conf. Inetd escucha las conexiones en los puertos bien conocidos e inicia el demonio adecuado cuando se requiere alguna conexión.

Para cambiar los demonios administrados por inetd, se debe ajustar el archivo de configuración de manera apropiada y enviarle una señal HUP a inetd:

```
ps ax | grep inetd # obtener el PID
kill -HUP PID
```

Cada línea del archivo de configuración de inetd, contiene información para administrar un servicio particular:

- Nombre del servicio - del archivo /etc/services
- Tipo de socket - (stream, dgram, raw, rdm, seqpacket)
- protocolo - del archivo /etc/protocols
- wait/nowait - se especifica para sockets de datagrama si inetd debe esperar hasta que la instancia actual del servidor sea procesada antes de atender otra petición en ese socket.
- Usuario - el nombre del usuario que será el encargado de correr el servicio (no necesariamente root).
- programa servidor - la ruta del programa que deberá ser arrancado por inetd.
- argumentos al servidor - argumentos al programa servidor, donde se incluye

también el nombre del programa.

Los campos se separan, ya sea por blancos o tabuladores; los comentarios empiezan con el carácter '#'. Inetd es capaz de procesar algunos servicios de manera autosuficiente; en estos casos, el nombre del programa servidor se especificará como "internal".

El formato de un archivo /etc/inetd.conf se muestra a continuación:

```
# @(#)inetd.conf    3.5 LAI System V.3 STREAMS TCP/IP  source
shell           stream tcp    nowait          root          /etc/rshd rshd
login           stream tcp    nowait          root          /etc/rlogind rlogind
telnet          stream tcp    nowait          root          /etc/telnetd telnetd
ftp             stream tcp    nowait          root          /etc/ftpd ftpd
exec            stream tcp    nowait          root          /etc/rexecd rexecd
tftp            dgram  udp    wait            daemon        /etc/tftpd tftpd
finger          stream tcp    nowait          daemon        /etc/fingerd fingerd
ntalk           dgram  udp    wait            root          /etc/talkd talkd
echo            dgram  udp    -               root          internal
chargen         stream tcp    -               root          internal
#chargen        dgram  udp    -               root          internal
smtp            stream tcp    nowait          mmdf          /usr/mmdf/chans/smtpd
```

9. El archivo /etc/services

El archivo /etc/services es un registro de programas y puertos conocidos. Si dos programas en máquinas diferentes desean intercambiar información, las computadoras tienen que coordinarse para llevar a cabo este intercambio, y para lograrlo utilizan un puerto bien conocido. Por ejemplo, el servicio para transferencia de archivos, /etc/ftpd, utiliza el puerto 21 cada vez que corre. Cuando el cliente ftp de una máquina se conecta al servidor de otra, utilizan el puerto 21 del protocolo TCP.

Cualquier programa que desee utilizar un puerto deberá registrarse en el archivo /etc/services. Todos los servicios que administra inetd deben estar registrados en el archivo /etc/services. Cada línea en el archivo describe un programa, el puerto que utiliza y el protocolo de bajo nivel con el cual se implementa la aplicación. El formato del archivo se describe a continuación:

- Nombre oficial del servicio
- número de puerto/nombre de puerto
- aliases

los campos van separados por blancos o tabuladores; los comentarios empiezan con el carácter '#'. Una parte correspondiente al archivo /etc/inetd.conf mostrado anteriormente es:

```
#      @(#)services   3.5 Lachman System V STREAMS TCP  source
#      @(#)services    1.16 (Berkeley) 86/04/20
#
# Network services, Internet style
#
echo          7/tcp
echo          7/udp
discard       9/tcp    sink null
discard       9/udp    sink null
systat        11/tcp   users
daytime       13/tcp
daytime       13/udp
netstat       15/tcp
qotd          17/tcp   quote
ftp           21/tcp
telnet        23/tcp
```

10. El archivo /etc/protocols

Si el usuario crea su propio protocolo utilizando la interface de sockets raw, tendrá que listarlo en el archivo /etc/protocols, de manera que inetd sea capaz de manipular los demonios que utilizan dicho protocolo. Cada línea del archivo describe algún protocolo utilizado en la red:

- Nombre oficial del protocolo
- número de protocolo
- aliases

El formato es el mismo que el mencionado anteriormente. Un ejemplo se muestra a continuación:

```
#
# @(#)protocols    3.2 LAI System V.3 STREAMS TCP/IP  source
#
# Internet (IP) protocols
#
ip      0       IP      # internet protocol, pseudo protocol number
icmp    1       ICMP    # internet control message protocol
ggp     3       GGP     # gateway-gateway protocol
tcp     6       TCP     # transmission control protocol
pup     12      PUP     # PARC universal packet protocol
udp     17      UDP     # user datagram protocol
```

11. Network File System (NFS)

NFS fue introducido por Sun Microsystems en 1985. La especificación de NFS es de dominio público, y se ha convertido en un estándar de facto que ha sido implementado por la mayoría de los proveedores de UNIX que soportan el sistema de red BSD.

NFS proporciona un sistema de archivos distribuido que es casi transparente a los usuarios. NFS además tiene una cualidad importante en el aspecto de integridad: si el servidor deja de funcionar, el cliente no pierde ninguna información. El cliente espera a que el servidor se reinicialice como si nada hubiera pasado.

11.1 El comando mount

El comando mount se ha modificado bajo NFS, de manera que pueda utilizar una notación extendida de las rutas de directorios, está notación es:

nombre del nodo:directorio

el cual indica el directorio en el nodo señalado. Por ejemplo, desde la máquina ehecatl el comando

```
/etc/mount kelem:/usr/exports/manodt /usr/man
```

haría que el directorio /usr/exports/manodt en la máquina kelem apareciera como un directorio local bajo /usr/man en ehecatl. Los comandos df o mount mostrarían todos los sistemas de archivos, ya sean locales o remotos.

```
$ df
/        (/dev/root      ):    2184 blocks    10919 i-nodes
/usr/man  (kelem:/usr/exports/manodt):   139850 blocks  0 i-nodes

$ mount
/ on /dev/root read/write on Sat Jun 20 22:45:25 1992
/usr/man on kelem:/usr/exports/manodt read/write on Sun Jun 21 15:48:39 1992
```

Existen varias opciones para mount que deberán considerarse cuando se montan sistemas de archivos remotos. Las páginas del manual de mount las describen en más detalle. Podemos señalar, no obstante, que las opciones importantes a considerar serían:

+ soft.- Regresa error si el servidor no contesta.

+ hard.- Si el servidor no contesta sigue intentando la conexión (tal vez para siempre).

+ bg.- Hace el primer intento de conexión en foreground, si falla sigue en background.

+ rw.- Sistema de archivos de lectura escritura.

+ ro.- Sistema de archivos de sólo lectura.

Para montar de manera automática el sistema de archivos propuesto en el ejemplo al momento de inicializar el sistema, se deben hacer cambios en los archivos /etc/fstab en ehecatl y /etc/exports en kelem. De esta forma, el archivo /etc/fstab incluiría la siguiente línea:

kelem:/usr/exports/manodt:/usr/man:rw:0:0:nfs:soft,bg

y el archivo /etc/exports incluiría lo siguiente:

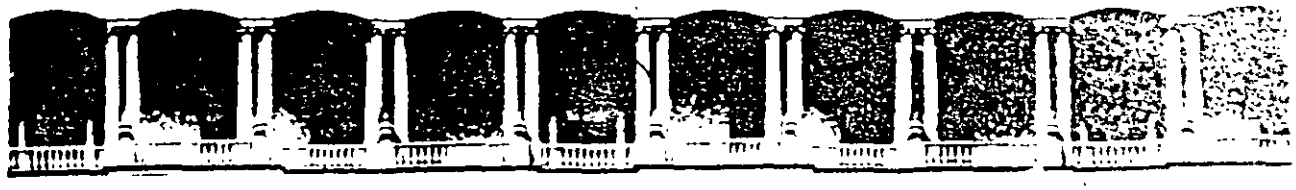
/usr/exports/manodt ehecatl

además hay que verificar que los siguientes demonios estén corriendo:

+ portmap
+ mountd
+ nfsd
+ biod

mountd y nfsd necesitan correr únicamente en el servidor, en este caso kelem; portmap y biod deben correr tanto en el cliente como en el servidor. Mountd es usualmente manejado por inetd. Portmap, nfsd y biod se arrancan en el /etc/rc.local. Dependiendo de la carga esperada sobre el servidor se deben arrancar copias múltiples de estos procesos, generalmente de 4 a 8. A continuación se ilustran extractos del archivo /etc/rc.local que inicializan el servicio de disco.

```
(echo -n 'inicializando servicio rpc y de red:') > /dev/console
if [ -f /etc/portmap]; then
        /etc/portmap; (echo -n ' portmap') > /dev/console
fi
#
echo -n 'daemons NFS'
[ -f /etc/mountd -a -f /etc/portmap -a -s /etc/exports] && {
                /etc/mountd -i; echo -n ' mountd -i' > /dev/console
}
[ -f /etc/nfsd -a -f /etc/portmap ] && {
                /etc/nfsd 4 ; echo -n ' nfsd'  > /dev/console
}
[ -f /etc/biod ] && {
                /etc/biod 4 ; echo ' biod'     > /dev/console
}
```



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

SISTEMA OPERATIVO UNIX (PARTE II)

ADMINISTRACION DEL SISTEMA

C O M P L E M E N T O

SEPTIEMBRE - OCTUBRE 1997

Palacio de Mineria Calle de Tacuba 5 Primer piso Deleg. Cuauhtemoc 06000 Mexico, D.F. Apdo. Postal M-2285
Telefonos: 512-8955 512-5121 521-7335 521-1987 Fax 510-0573 521-4020 AL 26

ADMINISTRACION DE PROCESOS

UNIX es un sistema operativo multitarea, en donde aparentemente se pueden ejecutar varios procesos al mismo tiempo. Decimos que aparentemente, ya que el sistema operativo solamente atiende a un procesos en pequeños intervalos de tiempo **time slice** (generalmente cada 20 milisegundos); sin embargo la rapidez con que atiende a cada uno de los procesos da la impresión de que todos los procesos se ejecutan al mismo tiempo.

En UNIX se puede tener cierto control sobre los procesos del usuario, es decir se puede monitorear su desempeño, controlar el tiempo de CPU que utiliza, suspender o terminar un proceso, etc.

1. Proceso

Hasta este momento hemos manejado el termino proceso indistintamente; pero ¿que es un proceso?

Un proceso consiste de un espacio en memoria y un conjunto de estructuras de datos que lo identifican. El espacio en memoria que el sistema operativo marca para uso exclusivo de un proceso contiene espacio para el código del programa, para las variables que el proceso utiliza y para el stack.

En sistemas de memoria virtual, el espacio en memoria asignado al proceso puede ser toda o casi toda la memoria física disponible para el sistema operativo.

Las estructuras de datos, que son almacenadas en el núcleo del sistema operativo, contienen información como:

- El mapa de direcciones de memoria.
- El estado.
- La prioridad de ejecución.
- Registro de los recursos utilizados.
- El dueño del proceso.

Muchos de los parámetros asociados a un proceso afectan directamente la ejecución de este: archivos que puede accesar, salida de datos, etc. estos se encuentran almacenados ya sea en el kernel en las estructuras de datos o bien en su espacio de memoria. A continuación se explican los más importantes.

Identificador de Proceso (PID)

El identificador de proceso (PID, *Process IDentification*) es un número único que el núcleo asigna a un proceso para su identificación. El número es asignado conforme los procesos se van creando.

Identificador del Proceso Padre (PPID)

El Identificador del Proceso Padre (PPID, *Parent Process IDentification*) es el PID del proceso que creó el proceso en cuestión.

Identificador del Usuario (UID)

El Identificador del Usuario (UID, *User IDentification*) es un número con el que se identifica al usuario que creo el proceso. Este usuario es el dueño del proceso y es el único que puede cambiar los parámetros de este.

Efectivo Identificador de Usuario (EUID)

El Efectivo Identificador de Usuario (EUID, *Effective User IDentification*) es el UID utilizado para determinar los recursos que puede accesar el proceso. Para la mayoría de los procesos el UID y el EUID son los mismos, la excepción son los programas que utilizan el setuid.

Prioridad

La prioridad de un proceso determina el tiempo de CPU que dicho proceso puede utilizar. El CPU determina el siguiente proceso a ejecutarse en base a las prioridades. Para sistemas BSD las prioridades de un proceso están en el rango +19 a -19 y en sistemas ATT están en 40 a -20, en donde las prioridades más altas son las negativas.

Todos los procesos de usuario tienen un prioridad por omisión. El dueño de un proceso solo puede disminuir la prioridad de este; por otra parte, un proceso solo puede cambiar su prioridad de acuerdo a como lo pueda su dueño. Un proceso hereda la prioridad del proceso que lo crea.

Terminal asociada

Todos los procesos en UNIX tienen asociada una terminal desde la cual se llevan a cabo las operaciones de entrada y hacia la cual se dirigen las salidas del proceso (esto a menos que haya redireccionamiento).

Estados de un proceso

Debido a que un proceso no se encuentra en ejecución en todo momento, se definen cinco posibles estado para un proceso:

- Ejecución: un proceso en ejecución es un proceso que ha adquirido tiempo de CPU en ese instante, así como también todos los recursos necesarios para entrar en operación.

- Sleeping: un proceso en estado sleeping es aquel que espera la ocurrencia de un evento; mientras tanto, el proceso no puede solicitar tiempo de CPU.

- Swapp: los procesos en estado de Swapp son aquellos que han sido transladados de memoria a disco. Este estado de un proceso se genera principalmente en sistemas que no tienen memoria virtual.

- Zombie: los procesos Zombie son procesos que no tienen nada; pero por alguna razón el kernel guarda información de ellos.

- Stop: un proceso en estado de Stop es aquel que ha sido marcado por el núcleo para que no pueda entrar en ejecución a menos que se le mande una señal de continuación. Los procesos son enviados a este estado cuando se les manda una señal de STOP.

2. Señales

Las señales son el medio por el cual se pueden controlar los procesos o bien, se puede llevar a cabo una comunicación entre ellos. Cuando un proceso recibe una señal, pueden suceder dos cosas: si el proceso cuenta con una rutina o manejador para esa señal, esta es ejecutada; de otra forma, el núcleo proporciona un manejador para esta señal. Cuando existe una manejador para una señal se dice que la señal se "atrapa".

Para prevenir que deliberadamente se envíen señales a un proceso, este puede ignorar o bloquear ciertas señales. Existen dos señales que no pueden ser atrapadas, ignoradas o bloqueadas: KILL y STOP.

La siguiente tabla muestra las señales existentes

Número	Nombre	Descripción
1	SIGHUP	Hangup
2	SIGINT	Interrupción
3	SIGQUIT	Terminación
*	SIGILL	Instrucción ilegal
*	SIGTRAP	Trace trap
*	SIGIOT	IO Trap
*	SIGEMT	Excepción aritmética
9	SIGKILL	Destrucción
*	SIGBUS	Error de bus
*	SIGSEGV	Segmentation faul
12	SIGSYS	Llamada al sistema con argumentos no válidos
*	SIGALARM	Alarma
15	SIGTERM	Terminación de proceso
*	SIGURG	Socket in extremis
17	SIGSTOP	Detención
*	SIGTSTP	Keyboard stop
*	SIGCONT	Continuación
*	SIGCHLD	Estado del proceso hijo ha cambiado
*	SIGTTIN	Lectura no válida
*	SIGXCPU	Tiempo de CPU excedido
*	SIGXFSZ	Tamaño de archivo excedido
*	SIGLOST	Recurso dañado

* indica que el numero varia dependiendo del systema

Envío de señales

La comunicación entre procesos se puede llevar a cabo por medio del envió de señales. El comando **kill** permite enviar una señal a un proceso en particular. Para enviar una señal a un proceso, es necesario ser el dueño de dicho proceso, solamente el superusuario puede mandar señales a cualquier proceso. La sintaxis del comando kill es la siguiente:

kill [-señal] PID

La señal se puede especificar por su número o nombre, PID es el Identificador del Proceso al que se le envía la señal. Si no especifica señal, se envía la señal 15 (TERM).

3. Modificación de la prioridad de un proceso

La prioridad de un proceso puede ser cambiada para aumentar o disminuir el tiempo de CPU que recibe. Los usuarios normales solamente pueden decrementar la prioridad de sus procesos, el superusuario es quien puede cambiar la prioridad de cualquier proceso en cualquier sentido.

En sistemas BSD la prioridad de un proceso puede determinarse al momento de crear el proceso, con ayuda del comando **nice**, el cuál tiene la siguiente sintaxis:

nice [nivel] comando [argumentos]

Si no se especifica un nivel de prioridad se crea un proceso con prioridad 10. Si un nivel de prioridad se especifica, este se añade el nivel por omisión para determinar el nuevo nivel de prioridad. El nivel de prioridad por omisión es 0. Por ejemplo, para incrementar la prioridad a un proceso se deberá especificar un nivel negativo:

```
% nice -10 comando
%
```

Cuando se trabaja con csh, el comportamiento de nice varía un poco, en este caso el comando es opcional y el nivel de prioridad es interpretado relativo al nivel del proceso padre. Cuando no se especifica proceso en el comando nice, se cambia el nivel de prioridad para el proceso de shell actual (por lo tanto para sus procesos hijos). Cuando no se especifica un nivel de prioridad, se incrementa en 4.

En sistemas BSD, la prioridad de un proceso también puede cambiarse una vez que este se ha creado; esto se logra con ayuda del comando **renice**. La sintaxis para el comando es la siguiente:

renice nivel [-p PID ...] [-g pgroup] [-u usuario]

El nivel especifica el nuevo nivel de prioridad. Con la opción **p** se especifican los PID's de los procesos afectados, la opción **g** sirve para indicar un grupo, con lo cuál se afectan los procesos que pertenecen al grupo especificado. Por último la opción **u** permite listar el usuario para el cual sus procesos serán afectados.

En sistemas AT&T, el comportamiento del comando nice es diferente. En este caso el nivel de prioridad se interpreta de acuerdo al nivel de prioridad actual del proceso. Por ejemplo:

```
% nice -5 nomina
%
```

incrementa el nivel de prioridad en 5. Si no se especifica nivel, este se incrementa en 10. Solamente el superusuario puede bajar el nivel de prioridad (en realidad se incrementa la prioridad de los procesos):

```
% nice --10 nomina
```

Cuando se especifican niveles de prioridad de -100 o menos, los procesos se ejecutan en tiempo real, es decir se les asignan todo el tiempo de CPU.

4. Procesos en paralelo

Cada vez que se proporciona un comando al shell, este lo interpreta, lo ejecuta y cuando termina devuelve el control al usuario para que este pueda proporcionar un segundo comando. En realidad cada vez que se proporciona un comando al shell, este una vez que lo interpreta crea un proceso hijo para ejecutar el comando.

UNIX es un sistema operativo multitarea, por lo que se pueden mandar a ejecutar procesos al mismo tiempo (aunque en realidad no sucede así). La forma de crear procesos en paralelo es con el terminador &. Cuando se finaliza un comando con &, el shell crea un proceso hijo como es usual, pero no espera a que este termine, en lugar de ello, el shell muestra el PID del nuevo proceso y de inmediato muestra el indicador para otro comando:

```
% nomina &
2764
%
```

Si el proceso en paralelo requiere de alguna entrada, se le envía una señal de STOP. La salida generada por el proceso en paralelo siempre es enviada a la salida estándar mientras el usuario que arranco el proceso continué en sesión.

Cuando el usuario que inició el proceso paralelo termina su sesión de UNIX, el proceso paralelo (hijo del shell de inicio del usuario) es adoptado por el proceso *init* (cuyo PID es 1). Cuando es adoptado el proceso, este sigue conservando todos sus parámetros, salvo que ya no tendrá una terminal asociada y su PPID será 1. Por otra parte, la salida generada por este proceso se perderá.

En sistemas AT&T cuando un shell es terminado se envía una señal de HUP a todos sus procesos hijos, incluyendo los que fueron ejecutados en paralelo. Si se desea que un proceso paralelo continué después de que termina el shell desde el cuál se ejecutó, se deberá utilizar el comando **nohup** de la siguiente forma:

nohup comando &

Este comando ignora la señal NOHUP.

5. Monitoreo de procesos

El comando **ps** permite obtener información para los procesos existentes en el sistema. El comando ps incluye opciones, las cuales pueden variar de sistema a sistema.

Las versiones de BSD y ATT difieren en sus opciones, pero proporcionan básicamente la misma información.

En ambas versiones ps sin opciones proporciona una lista de los procesos del usuario que ejecuta el comando.

En sistemas BSD ps con las opciones **-aux** o **-axl** muestran los siguientes campos de información:

USER	Dueño del proceso
PID	Identificador del proceso
%CPU	Porcentaje de CPU utilizado por el proceso
%MEM	Procentaje de memoria real utilizada por el proceso
VSZ	Tamaño virtual del proceso en kbytes
RSS	Tamaño residente en memoria(número de paginas de 1kbyte)
TT	Identificador de la terminal asociada al proceso
STAT	Status del proceso:

R	running	D	waiting
I	sleeping (<20 seg)	S	sleeping (> 20 seg)
T	stopped	Z	zombie

Flags:

L	Algunas páginas bloqueadas
s	Proceso de una sesión
W	Proceso en swap
+	proceso en foreground en su terminal asociada

START	Tiempo en el que se comenzo a ejecutar el proceso
TIME	Tiempo de CPU consumido
COMMAND	Comando y argumentos que generaron el proceso

En sistemas ATT ps con las opciones **-ef** o **-elf** muestran los siguientes campos de información:

F	Flags:			
	00	terminado	01	proceso del sistema
	02	debugeado	04	Stop cuando se comenzo el debug
	10	en memoria		
	Status:			
	O	en ejecución	S	sleeping (esperando un evento)
	Z	zombie	I	creandose
	R	elegible para ejecución		
UID	Identificador del dueño del proceso			
PID	Identificador del proceso			
PPID	Identificador del proceso padre			
C	Utilización del CPU			
PRI	Prioridad			
NI	Valor nice			
ADDR	Dirección de memoria del proceso			
SZ	Tamaño en páginas de memoria del proceso			
WCHAN	Dirección del objeto que el proceso esta esperando			

6. DAEMONS

Un daemon es un proceso en paralelo que lleva a cabo tareas de administración del sistema.

Un daemon es un programa, no es parte del kernel del sistema operativo.

Muchos daemons se levantan a tiempo de boot y continuan en ejecución mientras no se de de baja el sistema. Otros daemons se leventan cuando se necesitan y estan en ejecución solo el tiempo necesario para hacer las tareas que le corresponden.

7. PROCESOS PERIODICOS

Una de las alternativas para tener un mejor control de las tareas y procesos de administración en un sistema operativo, lo es el automatizar la ejecución de algunos procesos.

La ejecución periódica de procesos se lleva a cabo por el daemon **cron,** el cuál comienza a ejecutarse en cuanto se leventa el sistema y permanece así mientras el sistema lo este. El cron lee uno o más archivos de configuración, los cuales contienen los comandos y los tiempos en los que estos se deberán ejecutar. Los comandos son ejecutados por el shell sh.

En sistemas BSD el archivo de configuración para el cron, llamado " crontab" (cron table), generalmente es /usr/lib/crontab o /etc/crontab.

En sistemas ATT se cuenta con un directorio para los archivos de configuración, en donde existe un crontab por cada usuario. El comando crontab se utiliza para guardar archivos en este directorio.

Las versiones más viejas de cron hecen lecturas periódicas del crontab y ejecutan todos los comandos que se deben ejecutar desde la última lectura del crontab. Las versiones recientes hacen la lectura del crontab y obtienen el siguiente comando a ejecutarse para esperar en estado de sleep hasta que el tiempo de ejecución del comando llegue.

En algunos sistemas es necesario enviar una señal al cron para indicarle que debe volver a leer el crontab ya que este ha sido modificado.

En algunos sistemas se mantiene un archivo, normalmente */var/cron/log* o */usr/lib/cron/log*, en donde se mantiene el registro de los comandos que se ejecutan por el crfon y el tiempo en que fueron ejecutados.

En sistemas ATT se puede configurar que usuarios pueden hacer uso del cron y que usuarios no lo pueden hacer, para ello se utilizan los archivos *cron.allow* y *cron.deny*.

Formato de los archivos crontab

El número de crontabs en los sistemas UNIX varia, pero el formato de estos archivos es el mismo en todos los sistemas.

Cada línea de un crontab representa un comando que se ejecuta periódicamente. La línea esta formada por seis campos separados por blancos:

minuto hora día mes día_de_la_semana usuario comando

minuto, hora, día, mes y *día_de_la_semana* proporcionan información acerca del tiempo de ejecución de los comandos. Su interpretación es la siguiente:

minuto	minuto de la hora	0-59
hora	hora del día	0-23
día	día del mes	1-31
mes	mes del año	1-12
día_de_la_semana	día de la semana	1-7

Cada campo puede contener:

- * indicando cualquiera
- un solo valor
- valores separados por comas, indicando una lista de valores
- un rango de valores en la forma número-número

Página intencionalmente blanca.

CAPITULO IV

El sistema de Archivos

La estructura que permite organizar la información de UNIX se conoce como **sistema de archivos**. El sistema de archivos es sencillo y fácil de utilizar, no impone estructura alguna a los archivos, ni asigna significado a su contenido, el contenido de los archivos depende únicamente del programa o utilería que lo utilice como entrada.

1. El sistema de archivos

Un archivo en UNIX puede contener cualquier tipo de caracteres y su estructura depende del uso que se le de. Puesto que los tipos de archivos no son determinados por el sistema de archivos, el núcleo no puede decirnos cuál es el tipo de un archivo en particular, ya que no lo conoce.

El sistema de archivos de UNIX tiene las siguientes características:

- Una estructura jerárquica arborescente, en donde el nodo principal es el directorio llamado raíz (representado como "/") y cada uno de los niveles del árbol representan directorios.

- Consistencia en el manejo de archivos.

- Protección para archivos.

- Manejo de los dispositivos periféricos por medio de archivos.

La figura 1 muestra la representación de árbol de un sistema de archivos común en UNIX. El nombre de un directorio o archivo contiene toda la ruta que se sigue para llegar a él, comenzando con el directorio principal ("/"). Nombres de directorios son: `/etc`, `/usr/users/alu01/bin`, `/bin`, `/usr/spool`.

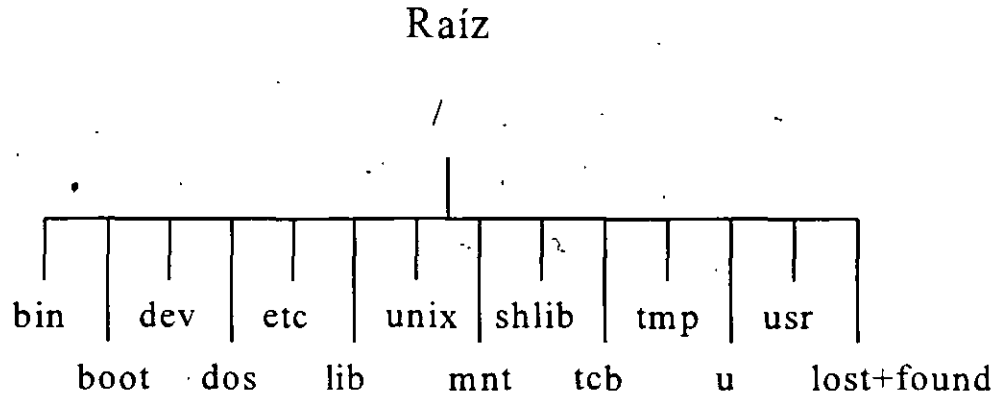


Figura 1 El sistema de archivos

Los directorios que se encuentran debajo del directorio raíz normalmente se encuentran en todas las implementaciones de UNIX, ésta figura en particular muestra el directorio raíz de SCO-UNIX. En la siguiente tabla se da una descripción general de algunos de los archivos y directorios más comunes e importantes.

Directorios estándares y su contenido	
ruta	Contenido
/	directorio raíz del sistema de archivos.
/bin	utilerías y comandos estándar.
/dev	archivos de dispositivos.
/etc	archivos de administración y usos diversos
/lib	bibliotecas.
/usr	sistema de archivos del usuario.
/sys	área para cons. del kernel. (BSD)
/tmp	archivos temporales.
/5bin	comandos de System V para compatibilidad con BSD
/unix	archivo ejecutable del sistema operativo.
/etc/adm	directorio con información administrativa.
/etc/init	comando que crea el primer proceso en el procedimiento de boot.
/etc/passwd	archivo de usuarios válidos del sistema.
/etc/shutdown	comando para dar de baja el sistema.
/etc/reboot	comando para dar de baja el sistema.
/usr/etc	comandos de administración (SUN,HP-UX).
/usr/adm	archivos de registro del sistema.

/usr/bin	archivos ejecutables
/usr/games	directorio de programas de juegos.
/usr/include	archivos de encabezados para C.
/usr/lib	archivos de soporte para programas de unix.
/usr/dict	diccionario (BSD).
/usr/man	páginas del manual en línea.
/usr/spool	colas de impresión, correo, uucp, etc.
/usr/tmp	más espacio temporal.
/usr/src	código fuente de los comandos estándar.
/usr/ucb	utilerías y programas de Berkeley (BSD).
/usr/lbin	comandos binarios locales (ATT).
/usr/local	Software local (BSD)
/usr/local/adm	archivos de registros de la auditoría.
/usr/local/bin	ejecutables del sistema local.
/usr/local/lib	archivos de soporte.
/usr/local/etc	comandos de mantenimiento del sistema.
/usr/local/src	código fuente de los comandos locales.
/usr/etc/yp	comandos de NIS (HP-UX)
/etc/yp	comandos de NIS (SCO)
/tcb	Trusted Computing Base. Información confidencial de passwords y auditoría
/mnt	directorio vacío para montar temporalmente sistemas de archivos.

Archivos y directorios importantes.

2. Directorios y nombres de archivos.

Para nombrar a un archivo se deben tomar en cuenta ciertas convenciones:

- La longitud máxima del identificador es de 255 caracteres para las versiones de UNIX basadas en BSD y de 14 para las basadas en AT&T.

- Puede contener cualquier carácter. Se recomienda utilizar solamente letras, números y los caracteres "." y "_".

- Nombrar un archivo con espacios en blanco o los caracteres *, ?, [,], -, $, ', `, ", &, y ! puede acarrear problemas, ya que estos tienen significado especial para el intérprete de comandos del sistema (shell).

- Se hace distinción entre letras mayúsculas y minúsculas.

- No existen convenciones para los nombres de archivos, por lo tanto se puede asignar o no una extensión a un tipo de archivo.

- No existen diferentes versiones de archivos ni tampoco archivos de respaldo.

- Cuando un nombre de archivo comienza con "." (punto), el archivo tiene significado especial para el sistema y generalmente se encuentra escondido.

Como ya se ha mencionado, UNIX no impone estructura alguna a los archivos, todos son tratados de la misma forma ya que el sistema los ve solamente como una secuencia de bytes. Sin embargo, podemos determinar el tipo de un archivo con ayuda del comando `file`. Este comando examina el archivo y de acuerdo a el tipo de caracteres que contenga determina si se trata de un archivo de texto, de un archivo con código ejecutable, de un directorio o de otro tipo.

3. Tipos de Archivos

Entre BSD y ATT, existen ocho tipos de archivos en UNIX.

- Archivos Regulares
- Directorios
- Archivos de dispositivo tipo carácter
- Archivos de dispositivo tipo bloque
- Unix-domain sockets (BSD)
- Entubamientos (FIFO's) (ATT)
- Ligas duras
- Ligas simbólicas

Archivos Regulares

El tipo más común de archivo, es el archivo regular. Los archivos regulares solo almacenan datos. Los datos pueden ser un programa, un archivo de texto, código fuente o cualquier otra cosa que pueda ser guardado en algún lugar. El kernel soporta acceso secuencial y aleatorio en todos los archivos regulares.

Directorios

Son como los archivos regulares en el sentido de que contienen un conjunto de datos, pero aquí el dato es restringido a ser una lista de otros archivos, el contenido del directorio. No existe limitaciones en el tipo de archivo que el directorio puede contener. Los sistemas BSD soportan archivos con longitud variable, y además tienen una estructura más compleja que los directorios de los sistemas ATT. BSD y recientes versiones de ATT ofrecen rutinas en librerías para ayudar a leer directorios; esto puede ser realizado a través de la llamada al sistema `read()` en los sistemas ATT antiguos.

Archivos de dispositivos tipo bloque y carácter

Los programas en UNIX se comunican con los dispositivos de Hardware a través de dos tipos especiales de archivos (device files), llamados archivos de dispositivos tipo bloque y carácter. Cuando un kernel de UNIX es construido módulos de código llamados manejadores de dispositivos (device drivers) son incluidos para conocer los detalles de la manera de comunicarse con dispositivos

particulares como los discos duros, flexibles, cintas, tarjetas de red, impresoras, etc...

El manejador de un dispositivo en particular oculta todos los detalles del manejo del dispositivo al kernel y al usuario final, presentando así, una interface de comunicación estándar que da la impresión de que se esta trabajando con un archivo regular.

Cuando el kernel recibe un requerimiento de manipulación de un archivo tipo carácter o tipo bloque, él no sabe como manipularlo y pasa el requerimiento directamente al manejador del dispositivo (device driver).

Los archivos especiales o archivos de dispositivos se identifican de la siguiente forma:

- Cada archivo especial tiene un número de dispositivo mayor y uno menor (major and minor device number)
- El major number identifica al manejador del dispositivo que necesita el kernel para acceder al dispositivo.
- El minor number significa un parámetro dependiente del manejador del dispositivo usado típicamente para diferenciar entre diversos tipos de dispositivos soportados por el manejador, o distintos modos de operación.

Major number

```
brw------- 2 sysinfo sysinfo 1 , 0   Jun 13 10:25 /dev/hd00
brw------- 2 sysinfo sysinfo 1 , 0   Jun 13 10:25 /dsk/0s0
brw-rw-rw- 5 bin     bin     2 , 4   Jun 13 10:05 /dev/fd048
brw-rw-rw- 5 bin     bin     2 ,52   Jun 13 10:05 /dev/dsk/f096
brw-rw-rw- 5 bin     bin     2 ,61   Jun 13 10:05 /dev/fd1135ds18
brw-rw-rw- 5 bin     bin     2 ,37   Jun 13 10:05 /dev/fd1135ds9
crw-rw-rw- 1 bin     terminal0 ,0    Jun 13 10:05 /dev/tty01
crw-rw-rw- 1 bin     terminal5 ,128  Jun 13 10:05 /dev/tty1A
crw-rw-rw- 1 bin     terminal5 ,0    Jun 13 10:05 /dev/tty1a
crw------- 2 lp      bin     6 , 0   Mar 20 14:58 /dev/lp0
```

Minor number

Los archivos de dispositivo tipo carácter tiene las siguientes características:

- Por convención, son guardados en `/dev` o en subdirectorios como `/dev/dsk`
- Utilizan un i-nodo pero no bloques de datos.
- Representan dispositivos en los que pueden leerse o escribirse cantidades arbitrarias de datos.
- Incluyen Sistemas de Archivos (FS), puertos seriales, puertos paralelos, terminales y cintas.
- También se les conoce como "raw devices" debido a que no manipulan la I/O. Los discos duros y los discos flexibles pueden ser accedidos de esta forma.

Los archivos de dispositivo tipo bloque tiene las siguientes características:

- Representan dispositivos que son escritos o leídos en bloques completos de 1024 Bytes a la vez, los cuales se manejan con un búfer.
- Sistemas de Archivos y Ram disks son ejemplos de archivos tipo bloque.
- Muchas aplicaciones prefieren los dispositivos a carácter para utilizar sus rutinas especializadas para el acceso a los datos.

Todos los archivos de dispositivos son creados con el comando `mknod` y son eliminados con el comando `rm`.

Unix-domain Sockets

Los sockets son conexiones entre procesos que les permiten comunicarse de una manera más rápida y fácil. Existen varios diferentes tipos de sockets en UNIX, muchos de los cuales involucran el uso de la red. Los sockets son locales a un nodo en particular y son referenciados a través de un objeto en el file system en vez de un puerto en la red. Los archivos de sockets son visibles a los demás procesos como entradas en el directorio, estas entradas no pueden ser leídas o escritas por procesos que no estén involucrados en la conexión del socket.

Los sockets en UNIX son creados con la llamada al sistema `socket()` y pueden ser eliminados con el comando `.rm` o con la llamada al sistema `unlink()`.

Entubamientos (ATT)

Similares a los sockets de UNIX, los entubamientos permiten la comunicación entre dos procesos ejecutándose en el mismo nodo. Los entubamientos pueden ser creados con el comando `mknod` y eliminados con el comando `rm`.

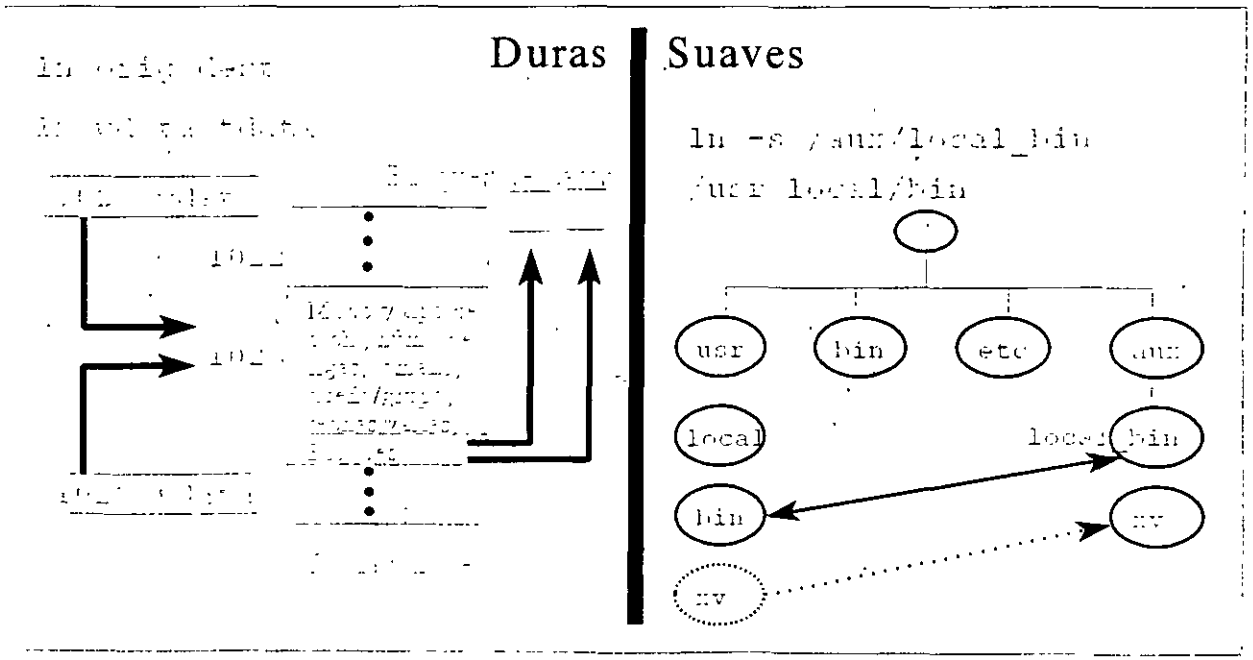
Ligas Duras

Una liga no es realmente un tipo de archivo, es un nombre adicional para otro archivo. Cada archivo tiene al menos una liga, usualmente el nombre bajo el cual fué originalmente creado. Cuando se hace una nueva liga hacia un archivo, un alias para este archivo es creado. Una liga es indistinguible del archivo al cual está ligado; para UNIX ambas cosas son lo mismo. UNIX mantiene un conteo de cuantas ligas apuntan hacia un archivo en particular y no libera el espacio que ocupa el archivo hasta que la última liga es eliminada. Como una liga dura es una conexión directa entre archivos, ésta no puede existir a través de file systems.

Ligas Simbólicas

Las ligas simbólicas son archivos que simplemente contienen el nombre de otro archivo. Cuando el kernel trata de abrir o pasar a través de la liga, su atención es directamente hacia el archivo que la liga simbólica apunta en vez de abrir la liga simbólica en sí. La diferencia entre las ligas simbólicas y las duras, es que las duras son una referencia directa, mientras que las simbólicas son una referencia a través de un archivo. Las ligas simbólicas son en sí archivos y, por lo tanto, tienen su propio dueño y permisos.

Las ligas simbólicas son creadas con el comando `ln -s`.



4. Permisos en los archivos

Cada archivo en UNIX tiene un conjunto de nueve bits de permisos asociados, que controlan quien puede leerlos, escribirlos o ejecutarlos. Y junto con otros tres bits que afectan la forma de ejecutarlos, constituyen los llamados bits de *modo,* que son los permisos totales del archivo. Los 12 bits de modo son guardados con otros cuatro bits, que contienen información acerca del tipo de archivo, en una palabra de 16 bits. Los cuatro bits tipo-de-archivo son fijados cuando el archivo es creado y no pueden ser modificados, pero los otros 12 bits pueden ser alterados por el dueño o por el super usuario usando el comando `chmod`. El comando `ls` es utilizado para examinar el valor de estos bits.

Los bits setuid y setgid

Los bits setuid y setgid tiene valores octales 4000 y 2000 respectivamente.

- Fijando el bit SUID (Set User ID) en un archivo binario, permite a los usuarios ejecutarlo con los permisos del dueño del archivo.

- Fijando el bit SGID (Set Group ID) en un archivo binario, permite a los usuarios ejecutarlo con los permisos del grupo del archivo.

- Fijando SUID o SGID en un archivo permite al programador crear un programa capaz de manipular archivos privados, pero no permite que los usuarios manipulen estos archivos directamente.

- Esto es, los usuarios que ejecutan comandos tales como `/bin/passwd` pueden escribir los cambios a el archivo `/etc/passwd`, el cual pertenece al grupo `auth`.

```
---x--s--x bin  auth ..... /bin/passwd
-rw-rw-r-- bin  auth ..... /etc/passwd
```

- Se debe ser el dueño del archivo, o ser *root*, para fijar el bit SUID o SGID. En SCO-UNIX se debe tener además la autorización `chmodsuid`.

- En algunos sistemas, tales com SunOS 4.0, el bit setgid puede ser fijado a un directorio para designar el grupo de default. Es decir, una vez fijado el setgid al directorio; cada archivo creado en él, tendrá como grupo el mismo grupo del directorio.

El sticky bit (bit pegajoso)

El bit con valor octal 1000 es llamado *sticky* bit. Cuando éste bit es fijado a un archivo ejecutable, el bit le dice al sistema operativo que el archivo se ejecutará con mucha frecuencia y por lo tanto deberá ser retenido en el área de swap aún cuando no sea ejecutado. Esto desperdicia área de swap, pero reduce el tiempo para ejecutar el programa dramáticamente.

- Algunos programas tienen activado el sticky bit por default; p.e. `vi`.
- Activando el sticky bit en un directorio, permite que sólo el dueño del archivo o root puedan borrarlo del directorio, el cual tiene permiso de escritura para el dueño, el grupo y otros.
- Indicado por una "t" en los permisos del directorio:

```
drwxrwxrwt 3 sys sys 2960 March 24 11:51 /tmp
```

- Sólamente root o el dueño del directorio pueden activar o remover el sticky bit.
- El sticky bit debería ser activado en todos los directorios públicos que permiten a los usuarios crear archivos en él.
- /tmp, /usr/tmp, y /usr/spool/uucppublic son ejemplos de directorios que son accedidos por todos vía el sticky bit.

`chsh` es un comando que permite al usuario cambiar su GCOS y/o Shell, éste existe en la mayoría de las implementaciones, pero no es SCO-UNIX, por lo que root deberá hacer un programa en C, y el ejecutable copiarlo o ligarlo a /users/local/bin/chsh y cambiarle "algunos" permisos.

```
# chmod 4755 chsh          activamos el SUID y fijamos el resto -rwsr-xr-x
# chmod u+s chsh         sólo activamos el SUID, el resto permanece igual
# chmod 2555 chsh                    activamos el SGUID -r-xrsxr-x
# chmod g+s chsh                          sólo activamos el SGUID
# chmod +s chsh                      activamos ambos -rsxrsxr-x
# chmod -s chsh                     eliminamos ambos -r-xr-xr-x
# chmod 1757 /users/tmp           fijo el sticky bit drwxr-xrwt
# chmod 0755 /users/tmp         elimino el sticky bit drwxr-xr-x
# chmod 0755 chsh               elimino los tres bits drwxr-xr-x
```

5. i-nodos

El kernel mantiene información acerca de los archivos existentes en una estructura llamada *i-nodo*. La tabla de i-nodos de un filesystem es formada cuando el filesystem es creado y su localización y tamaño nunca cambia. El sistema operativo mantiene una única lista de i-nodos en cada filesystem creado. Cada i-nodo (también llamada estructura i-nodo) contiene cerca de 40 piezas de información, pero muchas de éstas son útiles al kernel. Al administrador del sistema sólo le interesan las siguientes:

- Modo de acceso (permisos).
- Tipo de archivo (regular, directorio, liga. etc...)
- Número de ligas al archivo.
- El identificador del dueño (UID)
- El identificador del grupo (GID)
- Tamaño del archivo en bytes (o major y minor numbers, si se trata de un archivo especial)
- Fecha de último acceso.
- Fecha de última modificación.
- Fecha de último cambio de datos.
- Espacio para direccionar los bloques de datos.

  En algunas implementaciones se reservan 13, 10 directos y 3 indirectos.
  Y en otras se reservan 15, 12 directos y 3 indirectos.

Toda ésta información puede ser mostrada usando el comando `ls` con varias opciones. En BSD, `ls -lg` despliega un listado largo mostrando el tipo, modo, número de ligas, dueño, grupo, tamaño en bytes, fecha de la última modificación y el nombre del archivo, respectivamente. En ATT, `ls -l` produce el mismo resultado.

```
-rwxr-xr-x 1 root bin    57344  Sep 15 /bin/sh
crw-rw-rw- 1 root daemon 12, 1  Dec 14 /dev/ttya1
```

El primer carácter muestra el tipo de archivo, codificado conforme a la siguiente tabla. En el primer caso, una "-" indica que se trata de un archivo regular; en el segundo caso, una "c" indica que se trata de un archivo de dispositivo (device file) a carácter.

Codificación del tipo de archivo usado por `ls`			
Tipo de archivo	Símbolo	Creado por	Eliminado por
Archivo regular	"-"	editores, cp, etc..	rm
Directorio	"d"	mkdir	rmdir, rm -r
Dispositivo a carácter	"c"	mknod	rm
Dispositivo a bloque	"b"	mknod	rm
Unix domain socket (BSD)	"s"	socket(2)	rm
Named pipe (ATT)	"p"	mknod	rm
Liga dura (simbólica)	"l"	ln -s	rm

Con respecto a los permisos:

Bit deseado	Aspecto	Forma octal
Sólo lectura para el dueño, grupo y otros	`r--r--r--`	0444
Sólo escritura para el dueño, grupo y otros	`-w--w--w-`	0222
Sólo ejecución para el dueño, grupo y otros	`--x--x--x`	0111
SUID activado	`--s--x--x`	4111
SGID activado	`--x--s--x`	2111
sticky bit activado	`rwxrwxrwt`	1777
setuid activado, pero bit de ejecución no	`--S--x--x`	4011
sticky bit activado, pero bit de ejecución no	`--x--x--T`	1110

Con respecto al nombre del archivo:

Se pueden tener FS que soporten nombres cortos (máximo 14 caracteres) o nombres largos (máximo 255 caracteres). La longitud de 14 caracteres es el estándar. Todo depende de la implementación.

Características de la lista de i-nodos

- Se utiliza un sólo i-nodo para cada posible archivo.
- Un número fijo de i-nodos son creados en una tabla para cada sistema de archivos: éste número es igual al número de bloques lógicos del disco.
- Cuando un archivo es creado, el kernel remoueve un número i-nodo de la lista de i-nodos libres.
- El i-nodo 0 es usado para indicar cuando un archivo es removido de un directorio.
- El i-nodo 1 no se usa.
- El i-nodo 2 es el directorio raíz de cada sistema de archivos.

6. Creando File Systems

Se puede crear un sistema de archivos a través de una serie de comandos o por medio de alguna utilería como el `sam` de HP-UX o el comando `mkdev fs` de SCO-UNIX. Existen muchas razones por las cuales se desea añadir un nuevo FS.
Algunas de ellas son:

- Prevenir que el FS en uso llegue rápidamente a su máxima capacidad.
- Porque el FS en uso llegó ya a su máxima capacidad.
- Por que se desea separar físicamente porciones del FS.

Teóricamente, varios dispositivos pueden ser utilizados como File System, sin embargo, se recomienda usar el disco duro ya que un disco flexible, las cintas y cartuchos tienen una capacidad limitada y son lentos. Una cinta se utiliza normalmente para hacer backups de algún FS.

En algunas implementaciones se pueden crear archivos de dispositivo a carácter para crear el File Systems con el comando `newfs` y en otras con el comando `mkfs`. También se usa para crear el dispositivo a bloque para utilizarlos con el comando `mount` y `umount` y para acceder al FS.

mknod y divvy

El comando `mknod` tiene la siguiente sintaxis:

```
mknod /dev/[r]dsk/cxdysm c|b major 0xSSFDDD

donde:
cx   El número del controlador, en Series 700 puede ser:
     201  Core I/O
     4#   Bus EISA (4) y el número de slot (#)
dy   El número del manejador o la dirección del bus. Representa al
     dispositivo entero.
sm   El número de sección. Siempre es cero a menos que se use el
     software para división de discos duros (SDS) en series 700.
```

```
0x    notación hexadecimal

S     Número del módulo de bus del sistema
      1     Graphics
      2     Core I/O borad
      4     EISA Bus Adapter
      8     Procesador
      9     Memoria

S     EISA slot number, normalmente es 4

F     Function number
      1     SCSI
      2     LAN
      3     HIL
      4     Serial port 1 A
      5     Serial port 2 B
      6     Paralelo

DDD   Información específica del Driver

Por ejemplo:

El minor number 0x20133 se puede interptretar como:

0x    Notación hexadecimal
2     The core I/O card ocupa el slot 2
0     The core I/O card no no tiene slots
1     The SCSI function is number 1 on the core I/O card.
333   The driver-specific information es 0x333
```

Así, para crear el device file del disco duro SCSI con dirección de bus número 5 (suele ser el segundo disco del sistema, el raíz es el número 6).

```
mknod /dev/rdsk/c201d5s0    c    47    0x201500
mknod /dev/dsk/c201d5s0     b     7    0x201500
```

En SCO-UNIX se utiliza el comando `divvy` y el comando `mkdev fs`.

1) Se hacen las particiones con el primero de ellos:

```
divvy -b1 -c1 [-p0] [-v1]
```

- La tabla divvy lista los nombres de los File Systems y sus tipos. Si el FS es nuevo o si todavía existe, entonces incluye el número de división del FS y los bloques de inicio y final.
- -b: Es el número de dispositivo major para la interface de bloque.
- -c: Es el número de dispositivo major para la interface de carácter.
- -p: Es el dispositivo físico (disco duro)
- -v: Es el dispositivo virtual (partición)

2) Se instala el File System con el comando `mkdev fs`

- `mkdev fs` crea una entrada en `/etc/default/fylesys`, que determina cuando se monta el FS al momento de boot y cuando lo monta el usuario.
- `mkdev fs` también crea un directorio `lost+found` en la raíz del FS, el cuál debe estar creado para ser usado por `fsck`.

mediainit y badtrk

`mediainit`

El comando `mediainit` se utiliza en versiones AT&T para inicializar un disco. Entre las funciones que desempeña se encuentran las siguientes:

- Dar formato al dispositivo
- Verificar la integridad del mismo haciendo pruebas de lectura y escritura de patrones
- Detectar y marcar bloques dañados para una operación libre de errores
- Asignar un factor de INTERLEAVE

La sintaxis para el comando es:

```
mediainit [-vr] [-i interleave] char_dev_file
```

donde:

-v Verbose. Normalmente, mediainit manda a la salida de diagnóstico (stderr) sólo los mensajes de error fatal. La opción -v especifica que toda la información especifica del dispositivo sea mandada, a un bajo nivel de operación de mediainit, hacia la salida estándar (stdout).

-r Re-certificar. Forza a llevar a cabo una certificación completa del dispositivo, se haya o no certificado con anterioridad. Esto puede tomar una gran cantidad de tiempo extra.

-i interleave Factor de INTERLEAVE. Se refiere a la relación entre los registros lógicos secuenciales y los registros físicos secuenciales. La elección de un factor determinado puede tener un impacto sustancial en el rendimiento del disco. Si el disco requiere ser inicializado con un factor dado, pero éste no se especifica, mediainit provee uno apropiado, pero no necesariamente el óptimo.

Char_dev_file Nombre del archivo de dispositivo a carácter asociado con el dispositivo que será inicializado. mediainit termina su ejecución si no se tienen los permisos de lectura y escritura del archivo de dispositivo o si algún otro proceso tiene acceso al dispositivo en ese momento. También terminará su ejecución si el manejador para ese tipo de dispositivo no está configurado en el kernel. Esto previene una inicialización accidental del dispositivo raíz o algún volumen montado.

Luego entonces, algunos ejemplos de como usar mediainit son:

```
mediainit /dev/rdsk/c201d1s0
mediainit -v -f 3 -i 2 /dev/rdsk/3s0
```

`mediainit` destruye toda la información existente en el área que se inicializa. Por ello se recomienda realizar un back up de cualquier información existente antes de usar este comando. Muchos de los discos SCSI requieren un factor de INTERLEAVE específico para ser inicializados. Si no se tiene la certeza de conocer el factor debido, no se debe inicializar el disco. Se recomienda inicializar un disco antes de utilizarlo por vez primera. El tiempo de inicialización depende del tipo y tamaño del disco y puede variar de 10 a 20 minutos para discos de alto rendimiento y ser hasta de 45 minutos para discos más lentos.

`badtrk`

Busca en los discos duros sectores dañados y crea una tabla con ellos. Se usa durante la instalación de un nuevo filesystem, busca defectos sobre la superficie del disco, crea una tabla nueva de sectores dañados, imprime la tabla actual y añade y borra entradas en la tabla.

sintaxis

```
/etc/badtrk [ -sqtdn ] [ -f device]
```

-f device — Abre la partición asociada al dispositivo y lee en la tabla correspondiente a esa partición

-s arguments — Invoca a badtrk en forma no interactiva, provocando que busque los sectores dañados y los introduzca en la tabla de sectores dañados. Los argumentos especifican si la búsqueda es rápida o lenta, destructiva o no destructiva.

[q]uick
[t]horough
[d]estructive
[n]on-destructive

Se debe especificar q o t, seguida de d o n.

Si se usa en modo interactivo, despliega el siguiente menú principal:

```
1. Print Current Bad Track Table
2. Scan Disk (You can choose Read-Only or Destructive later)
3. Add Entries to Current Bad Track Table by Cylinder/Head Number
4. Add Entries to Current Bad Track Table by Sector Number
5. Delete Entries Individually From Current Bad Track Table
6. Delete All Entries From Bad Track Table

                Enter your choice or q to quit:
```

Se debe contestar con una opción numérica y, dependiendo de ésta, será requerida más información posteriormente.

newfs y mkfs

El comando `newfs` de HP-UX se usa para crear un File System nuevo en el volumen lógico del disco, con la infraestructura necesaria para soportar las estructuras de datos del propio FS. Este comando requiere de un archivo de dispositivo a carácter. Además , usa el archivo `/etc/disktab` para extraer de él los valores de los parámetros necesarios para crear el FS, en vez de teclear en la línea de comandos los parámetros y las opciones necesarias para un tipo de disco específico.

La sintaxis del comando es:

```
newfs [-F|-L|-S] [-v] [-n] [-mkfs_options] char_dev_file disk_type
```

donde:

[-F|-L|-S] HP-UX soporta longitudes de nombres largos de archivos, de 255 caracteres máximo y nombres cortos de 14 caracteres máximo; el uso de -F crea nombres de archivos con la misma longitud del sistema de archivos raíz.

[-v] Verbose.

[-n] Previene que el programa bootstrap sea instalado.

mkfs_options incluyen:

-s size Especifica el tamaño total del FS en bloques. Si no se especifica newfs calcula el tamaño máximo posible a ser usado en el disco.

-b block_size Especifica el tamaño básico del bloque en bytes para un archivo en el FS. Por default

se utilizan 8 Kbytes. Algunos tamaños pueden ser 4096, 8192, 16384,32768 y 65536.

-f frag_size Especifica el tamaño de fragmento del FS en bytes, el cual representa la menor cantidad de espacio del disco que puede ser usada por un archivo. Puede ser una potencia de 2 no menor que 1024 (DEV_BSIZE) ni menor que uno-ocho de los tamaños de bloque del FS. El default es 1024 bytes.

-m % minfree Indica el porcentaje de espacio libre reservado. Si el espacio libre llega a ser menor, el supurusuario será la única persona que podrá escribir en el FS. Se reserva para asegurar el buen funcionamiento de las políticas de almacenamiento de bloques de datos. Si se reduce hasta una cantidad menor al 10% se verá afectado el rendimiento en forma negativa. Por ello, el default es 10 por ciento.

-i bytes/inodo Especifica el número de i-nodos que serán creados. Para cada *x* número de bytes, un i-nodo será creado. Por default se maneja 1 i-nodo por cada 2048 bytes de espacio libre. Si se desean menos i-nodos, debe usarse un tamaño mayor en bytes.

Char_dev_file Es el archivo de dispositivo a carácter correspondiente al dispositivo en donde se quiere crear el FS.

Disk_type Es la etiqueta en el archivo `/etc/disktab` que corresponde al tipo de dispositivo.

Típicamente, los valores por default son adecuados para los propósitos del FS. No obstante, puede haber ocasiones en que es necesario especificar un valor distinto. Si una opción no es especificada, `newfs` calcula un valor apropiado o extraé el valor requerido del `/etc/disktab`. Antes de usar éste comando se debe:

a) Checar que el dispositivo no esté montado. Usar el comando `bdf`.

b) Encontrar una entrada a `/etc/disktab` para el manejador que contendrá el FS. Si no la hay, crearla.

Describiendo un ejemplo:

```
newfs /dev/rdsk/c201d2s0 hp2212A
newfs -b 4096 -f 2048 -m 5 -i 4096 /dev/rdsk/c201d5s0 hp2213A
```

se crea un FS, en un disco SCSI HP C2213A. El archivo de dispositivo es `/dev/rdsk/c201d5s0`, la etiqueta en `/etc/disktab` es hp2213A. Las opciones restantes sobrescriben los valores correspondientes para este tipo de disco en `disktab`. El tamaño de bloque es de 2048, hay 5% de espacio reservado y 4096 bytes de espacio del FS por i-nodo.

`newfs` añade las localidades del super bloque al archivo `/etc/sbtab`.

Para SCO-UNIX el comando correspondiente es `mkfs`, el cuál sirve para crear manualmente un File System. Su sintaxis es:

```
mkfs /dev/device size
```

Crea una estructura de FS completa: un superbloque inicializado, el bloque de boot, la lista de i-nodos y la tabla de bloques de datos (incluyendo el directorio raíz). Se debe tener especial cuidado con este comando, ya que `mkfs` borra el contenido previo del área que se usa para crear el nuevo File System.

7. Haciendo uso de File Systems

mount y umount

En SCO-UNIX se usán los comandos mount y umount para montar y desmontar filesystems.

Su funcionamiento es el siguiente:

`mount`

- un filesystem puede montarse en cualquier directorio.
- el directorio que servirá como punto de montaje debe estar vacío.
- Por seguridad solo root puede hacer uso de este comando.
- La ejecución puede fallar si hay algún otro filesystem montado en el punto en el cual se pretende hacer un montaje.
- El sistema no aceptará filesystems sucios para ser montados.
- Un filesystem se puede montar en otro que esté montado en ese momento.
- Un filesystem se puede guardar en algún tipo de almacenamiento removible como un diskette, un disco duro o CD-ROM para transportarlo fácilmente entre diferentes computadoras.
- Para montar un filesystem use:

```
mount [-r] device mount_point
```

donde: -r Opción de sólo lectura
device Nombre del dispositivo especial (filesystem, unidad de disco, etc).
mount_point Directorio en donde se desea montar el dispositivo especial

- Si se dá el comando `mount` sin parámetros, se obtiene una lista de todos los filesystems montados.

umount

- Previene acerca de un acceso a los datos no autorizado.
- Previene a los usuarios acerca de la forma de llevar a cabo cualquier trabajo en el filesystem especificado.
- Puede ser usado antes de limpiar el filesystem con `fsck`.
- Nunca se debe remover un filesystem antes de haberlo desmontado.
- No se puede desmontar un filesystem si hay alguien trabajando en él o si hay algún archivo abierto. `umount` regresa el mensaje: "Device busy".
- El filesystem raíz no puede ser desmontado.
- Solamente root puede ejecutar `umount`.
- Los archivos (datos y programas) pueden ser guardados en filesystems distintos y volverlos inaccesibles cuando son requeridos.
- Los filesystems desmontados son "invisibles".
- umount acepta el nombre de un dispositivo especial como único argumento:

```
umount /dev/rfd096
```

mnt y umnt

También en SCO-UNIX se manejan estos dos comandos como se ve a continuación:

- Cualquier usuario puede montar algún filesystem seleccionado usando el comando `mnt` siempre y cuando haya una referencia a dicho filesystem en el archivo `/etc/default/filesys`.

```
mnt [-tu] mount_point
```

- Un filesystem puede ser montado solamente en el punto de montaje especificado en `/etc/default/filesys`.

- Para examinar el contenido del archivo `/etc/default/filesys` se usa la opción -t:

```
mnt -t
```

- Un filesystem se desmonta (si se tiene autorización) con el comando `umnt`:

```
umnt mount_point
```

Montado automático

a) El archivo `/etc/fstab`

Cada sistema BSD tiene un archivo llamado `/etc/fstab` (/etc/filesystems, en algunos sistemas UNIX) que lista todas las particiones del disco disponibles para el sistema y su uso. Para particiones regulares, `/etc/fstab` también especifica el punto de montaje; las particiones que se usan para intercambio (swap) nunca son montadas, por lo que no tienen un punto de montaje.

Por ejemplo, el archivo `/etc/fstab` en "boulder" (una Vax) contiene las siguientes líneas:

```
/dev/hp0a:/:rw:0:1
/dev/hp0b:/:sw:0:0
/dev/ra0b:/:rw:0:0
/dev/ra0h:/usr:rw:0:2
/dev/ra1h:/usr/local:rw:0:2
/dev/hp0g:/usr/spool:rw:0:2
/dev/ra0a:/usr/spool/uucp:rw:0:4
/dev/ra0g:/faculty:rw:0:3
/dev/ra1g:/student:rw:0:3
/dev/ra1a:/tmp:rw:0:5
```

Cada línea tiene cinco parámetros (usualmente separados por dos puntos) que describen una partición sencilla. El primer campo es el nombre del archivo de dispositivo a bloque que representa la partición. El segundo campo contiene el nombre del punto de montaje para las particiones que contienen File Systems, o está vacío si la partición se usa para intercambio (swapping). El tercer campo contiene un código de dos letras para identificar el modo en que se debe de usar el FS. Los códigos que se pueden utilizar aparecen en la figura. El cuarto campo se utiliza cuando se usa el comando `dump` en la partición y el quinto campo afecta el orden en el cuál se checará la consistencia de las particiones por el comando `fsck` al momento de iniciar el sistema.

código de dos letra usado en el archivo /etc/fstab	
código	Significado
"rw"	Filesystem: montado para lectura y escritura
"ro"	Filesystem: montado para sólo lectura
"rq"	Filesystem: montado para lectura y escritura con quotas
"sw"	Partición de intercambio (swap), no se monta.

De la línea 6 del ejemplo anterior, podemos ver que la partición direccionada por el archivo `/dev/hp0g` contiene un filesystem que usualmente es montado en `/usr/spool` y que puede ser leído y escrito. Si no se usa el archivo `/dev/fstab` con el comando dump, el campo cuatro será cero. El quinto campo nos dice si el filesystem se debe checar en el segundo paso, en tándem con /dev/ra1h, el cuál es un archivo de dispositivo a bloque no a carácter. El nombre de la partición empieza con "r" sólo porque el nombre del disco es "ra". El archivo de dispositivo *raw* correspondiente para esta partición es /dev/rra1h.

Algunos sistemas, especialmente aquellos que soportan filesystems de red, cambian ligeramente el archivo `/etc/fstab`.

Este archivo es leído por los comandos `mount`, `umount`, `swapon` y `fsck`, por lo que es importante que la información contenida esté correcta y completa. Debe recibir mantenimiento por el administrador del sistema, ya que ninguno de los comandos mencionados hace algún tipo de ajuste en él.

`mount` y `umount` usan el archivo para figurar lo que se desea hacer si se especifica sólo la partición en la línea de comandos. Por ejemplo, el comando `mount /usr/local` puede ser completado con `mount /dev/ra1h /usr/local`. El comando `mount -a` monta todos los filesystems listados en `/etc/fstab`. Se ejecuta normalmente en `/etc/rc.local` en el momento de iniciarse el sistema.

mount lee el archivo en forma secuencial; por ello, un filesystem debe montarse siguiendo la partición de su filesystem padre, esto es, la línea para `/usr/local` debe seguir a la línea que contiene a `/usr`. El comando umount acepta una sintaxis semejante: no se puede desmontar un filesystem si sus archivos se encuentran abiertos o si el proceso mismo está en alguno de sus directorios.

b) El archivo /etc/default/filesys

```
bdev=/dev/root cdev=/dev/rroot \
     mountdir=/ mount=no fstyp=EAFS \
     fsck=no fsckflags= \
     desc="The root filesystem" \
     rcmount=no rcfsck=no mountflags=

bdev=/dev/d0s2 cdev=/dev/rd0s2 mountdir=/users rcmount=yes mount=no
fsckflags=-y
bdev=/dev/d0s3 cdev=/dev/rd0s3 mountdir=/pcs rcmount=yes mount=no
fsckflags=-y
```

c) El archivo /etc/checklist

```
/dev/dsk/c201d6s0 / hfs defaults 0 1 304 304
/dev/dsk/c201d5s0 /users hfs rw,suid,quota 0 2 31491 31485
/dev/dsk/c201d5s0 ..... swap end,pri=0 0 0 31491 832
/dev/dsk/c201d6s0 ..... swap pri=0 0 0 832
aries:/ /aries nfs rw,suid,soft,intr 0 0 0
odin:/www /www nfs rw,suid,bg 0 0 16388
rha:/cdrom /cdrom nfs rw,suid,bg 0 0 16388
```

d) El archivo /etc/fstab de Linux

```
/dev/hda2          /          ext2       defaults    1   1
/dev/hda1          /dos       msdos      defaults    1   1
/dev/sbpcd         /cdrom     iso9660    ro          1   1
none               /proc      proc       defaults    1   1
cronos.fi-b.unam.mx:/reas          /users        nfs      rw
brahm.fi-b.unam.mx:/reas/alum      /users/alum   nfs      rw
```

8. Checando y reparando File Systems

fsck

- Checa la integridad del filesystem y hace reparaciones.
- Checa el filesystem en fases.
- Debe ejecutarse en un filesystem desmontado.
- Puede ser ejecutado en forma interactiva y en shells-scripts

El comando existe para HP-UX y su sintaxis es:

```
fsck -p|-P [files_system]

fsck [ -b block# ] [-y|-n] [-q] [file_system]
```

si se corre sin opciones lo hará en modo interactivo. En este modo mostrará una pregunta cuando encuentre una inconsistencia y esperará por la respuesta.

Las opciones son las siguientes:

-p Checa el filesystem en paralelo con cualquier otro. Tomando ventaja del traslapamiento de las operaciones de I/O, para activar al filesystem tan pronto como sea posible.

-P Semejante a la anterior, excepto que no checa los filesystems que fueron desmontados correctamente (verifica el byte de limpieza). Esta forma es la más utilizada en los shells-scripts al momento de arrancar el sistema.

-y Causa que para cada pregunta del comando la respuesta sea "yes". Es posible que debido a una respuesta positiva algún dato sea borrado. En consecuencia, si se usa esta opción, el file system debió haber sido examinado con la opción -n primero para conocer las posibles consecuencias.

-n Responde "no" a todas las preguntas del comando. Esta opción nunca dá como resultado algún borrado de información, sin embargo, no toma alguna acción correctiva y, por ende, las inconsistencias no son resueltas. Se recomienda usarlo para conocer el estado del file system y posteriormente invocar el comando nuevamente para resolver las inconsistencias encontradas. Por ejemplo:

`fsck -n /dev/dsk/c201d1s0 | tee /usr/tmp/fsck.log`
resulta en un archivo temporal llamada `fsck.log` que contiene la salida redireccionada del comando fsck, para analizarla y determinar alguna acción correctiva.

-q Con esta opción se imprimen sólo aquellos mensajes que requieren una respuesta.

-b block# Esta opción le dice a fsck que use el block# como super bloque para el chequeo del file system. Es útil cuando el super bloque original está dañado o se ha perdido.

Se puede abortar el comando con Ctrl+C sin que se haya llevado a cabo alguna reparación ya que éstas se realizan sólo si el comando se ejecutó con éxito.

No se debe reinicializar el sistema a menos que fsck lo indique. Si se desea reinicializar el sistema debe usarse el comando `reboot -n` para que no se utilice la llamada `sync`, la que ocasiona que se escriba la información errónea en el disco.

df

- El administrador debe verificar que exista un mínimo de espacio e i-nodos libres en cada filesystem.
- Un filesystem funciona mejor si tiene al menos el 15% de espacio libre.
- `df` reporta el número de bloques libres. Cada bloque es de 512 bytes.

La sintaxis es:

```
df [-t] [-i]
```

-t reporta el número total de bloques e i-nodos, así como también el numero libre restante.

-i reporta el porcentaje de i-nodos usados, así como también el número de i-nodos usados y libres.

du

- Reporta el número de bloques (de 512 bytes) usados en el directorio especificado.

La sintaxis es:

```
du [-sa] directorio
```

-s Reporta la suma total de bloques utilizados
-a Reporta el número de bloques para cada archivo y directorio debajo del directorio especificado.

bdf

- Este comando existe en algunos sistemas AT&T, como es el caso de HP-UX.

La sintaxis es:

```
bdf [-i] [file_system]
```

-i Reporta la cantidad de i-nodos usados y libres
file_system Filesystem a analizar.

Ejemplo:

```
# bdf
Filesystem          kbytes    used   avail   capacity   Mounted on
/dev/dsk/c201d5s0   100047   84736    5306         94%   /
/dev/dsk/c201d6s0   483392  483392       0        100%   /users
```

Haciendo espacio

1. Instruya a los usuarios del sistema a eliminar archivos obsoletos.

2. Cortar el tamaño de los archivos temporales:

- Archivos de auditoría.
- Archivos de bitácora.
- Archivos core.
- Archivos de código fuente terminfo.
- `/etc/wtmp`
- `/usr/adm/messages`
- `/usr/adm/sulog`
- `/usr/adm/syslog`
- `/lost+found`
- `/tmp y /usr/tmp`

3. Use el comando para localizar y eliminar archivos, por ejemplo:

```
find / -type f -name core -print -exec rm {} \;
```

```
find / -type f -atime +365 -print -exec l {} \;
```

```
(find -type d -size +5 -print -exec l -d {} \;) | mailx -s
"directorios grandes" root
```

```
find /tmp /usr/tmp -atime +7 -exec rm -f {} \;
```

-size +5 encuentra los directorios que son más grandes que 5 (512 bytes) bloques.

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

SISTEMA OPERATIVO UNIX (PARTE II)

ADMINISTRACION DEL SISTEMA

C O M P L E M E N T O

SEPTIEMBRE - OCTUBRE 1997

Palacio de Mineria Calle de Tacuba 5 Primer piso Deleg. Cuauhtemoc 06000 Mexico D.F. APDO. Postal M-2285
Telefonos 512-8955 512-5121 521-7335 521-1987 Fax 510-0573 521-4020 AL 22

UNIX PARTE II

ADMINISTRACION DEL SISTEMA

MATERIAL ANEXO

31 de mayo de 1996

B-1. SLIDE: Terminal Interfaces

Terminal Interfaces

Series 700

- Built-in Core I/O RS-232-C Port A
- Built-in Core I/O RS-232-C Port B

Series 400

- Built-in RS-232-C interface
- HP 98626A RS-232-C interface
- HP 98628A Datacomm interface
- HP 98642A 4-Channel Multiplexer interface
- HP 98638A 8-Channel Multiplexer interface
- HP 98644A RS-232-C Serial interface

51436 B-1 243

Student Notes

Terminals connect to any supported RS-232-C interface.

On the Series 400, the HP 98628A Datacomm interface, the HP 98642A 4-Channel Multiplexer interface, or the HP 98638A 8-Channel Multiplexer interface are required for graphics terminals, unless their graphics capability will not be used.

Xterminals connect via the LAN. Xterminals use the industry-standard X Window System, Version 11, Release 4. Xterminals support OSF/MOTIF and come standard with a license to use HP's VUE. X station deliver the performance needed to display multiple complex graphic applications simultaneously. Pseud device files are needed for each window used in a session. The login process is controlled through `xdm` `vuelogin`.

The details of each connection will be discussed in detail on following slides.

B-2. SLIDE: The /etc/getty Process

The /etc/getty Process

- Normally invoked by `init` via the `/etc/inittab` file
- First command executed for each login
- Operates as follows:
  1. Displays the contents of `/etc/issue`
  2. Issues `login:` prompt
  3. Waits for you to type something then reads login name
  4. Establishes speed and case
  5. Invokes `/bin/login` passing it the login name you typed

`/etc/getty [ -h ] [ -t` *timeout* `]` *line* `[` *speed* `]`

`-t` *timeout* is the timeout value (seconds)
`-h` do not force line hangup after timeout
line is the tty line in `/dev`
speed is the label to speed definition in `/etc/gettydefs`

Student Notes

In a multi-user run-level, a `getty` process is running on each port in which a user can log in. The `getty` process is normally invoked by `init` according to an entry in `inittab`.

The purpose of `getty` is to set terminal options, print the contents of the `/etc/issue` file (if it exists), print a `login` prompt, wait for input to that prompt, and, following a response by a user, `exec /bin/login`.

You must supply `getty` with the device file name of the line on which it is to run. This device file should exist in the `/dev` directory.

Frequently, `getty` is invoked with a `-t` option followed by an integer value representing seconds. If this option is specified, `getty` opens a line and if nothing is typed in the number of seconds specified, `getty` exits.

getty may also be invoked with a -h option. This option is used primarily with modem connect ne
If specified, getty drops carrier when a user logs off. Another login prompt is not issued and th.. user
must dial in again to establish a new connection.

getty also has a speed option. If specified, it serves as a label in to the /etc/gettydefs file. The definition in /etc/gettydefs instructs getty at what speed to run, what to use as a login prompt, wha to set as initial tty line settings, and at what speed to try next if the initial speed is inappropriate. With HP-UX, a speed entry of 9600 should be used for terminals connected directly to the system. For dial-up ports, a label corresponding to the appropriate speed for the dial-up line should be used. If a speed valu is not designated, a default of 300 baud is used.

The *action* field for a getty entry in inittab is usually respawn. Thus, whenever the getty process terminates, usually when the user logs out, a "wake-up" signal is sent to init. init immediately forks a new getty process. The result is that another login: prompt appears on the terminal connected to tha port.

The actions of getty are extensive and complex. For more information, see *getty(1M)* in the *HP-UX Reference* manual.

Note

As shipped, /etc/inittab invokes /etc/getty only for the system console in run-level 2. If your system has additional terminals on which you wish to support logins, you mus add the appropriate getty entries to /etc/inittab. (SAM automatically creates these entries when you use it to add terminals.)

Note

If a user issues a telnet command or rlogin command the getty process is no. used. The network daemon starts /bin/login bypassing the getty process.

B-3. SLIDE: The /etc/gettydefs File

The /etc/gettydefs File

```
label# initial-flags # final-flags #login-prompt#next-label
```

label	Identifies the entry Matches against getty speed argument in /etc/inittab
initial-flags	Initial line and terminal settings Speed must be specified
final-flags	Final line and terminal settings Speed must be specified
login-prompt	Initial login-prompt printed on the terminal
Next-label	Entry to try next if "break" is typed

Student Notes

The file **/etc/gettydefs** contains information used by **getty** to set up speed and terminal settings for a line. Each entry in the **gettydefs** file contains a series of hash mark ("#") separated fields having the following format:

```
label# initial_flags # final_flags #login_prompt#next_label
```

The meaning of each field is as follows:

label — The string against which **getty** tries to match its second argument. It is often the speed, such as 1200, at which the terminal is supposed to run, but it need not be.

initial-flags — These flags are the initial `ioctl(2)` settings to which the terminal is to be set if a terminal type is not specified to **getty**. The flags that **getty** understands are the same as the ones listed in **/usr/include/sys/termio.h**. Normally only the speed flag is

required in the *initial-flags*. `getty` automatically sets the terminal to raw inp mod and takes care of most of the other flags. The *initial-flags* settings remain in effect u `getty` executes `/bin/login`.

final-flags — These flags take the same values as the *initial-flags* and are set just prior to when `getty` executes `login`. The speed flag is required. The composite flag SANE takes care of most of the other flags that need to be set so that the processor and terminal are communicating in a rational fashion. The other two commonly specified *final-flag* are TAB3, so that tabs are sent to the terminal as spaces, and HUPCL, so that the line hung up on the final close.

login-prompt — This entire field is printed as the *login-prompt*. Unlike the above fields where white space is ignored, white space is included in this field.

next-label — If this entry does not specify the desired speed, indicated by the users typing a break character, then `getty` will search for the entry with *next-label* as its *label* field and se up the terminal for those settings. Usually, a series of speeds are linked together into closed set. For example, 2400 is linked to 1200, which in turn is linked to 300, which finally is linked to 2400.

The syntax of the `gettydefs` file must be exact. Spaces within each entry must appear as shown on t slide and a blank line must follow each entry (apart from the very last line).

Two sample entries in the file `/etc/gettydefs` might look like these:

```
9600    # B9600 HPUCL SANE CS7 PARENB ISTRIP IXANY TAB3
        # B9600       SANE CS7 PARENB ISTRIP IXANY TAB3
        #login: #300

console# B9600 SANE CLOCAL CS7 PARENB ISTRIP IXANY TAB3 HUPCL
        # B9600 SANE CLOCAL CS7 PARENB ISTRIP IXANY TAB3 UUPCL
        # Console login: #console
```

For more information on the `gettydefs` file see *GETTYDEFS(4)*, *STTY(1)* and *TERMIO(7)* in the *HP-U Reference* manual.

Note

It is strongly recommended that after making or modifying `/etc/gettydefs`, it be ru through `getty` with the `-c` option (for check) to be sure there are no errors.

The following procedure is recommended:

```
# cd /etc
# cp gettydefs gettydefs.new
# vi gettydefs.new
```

B-4. SLIDE: Adding a Terminal

Adding a Terminal

What You Need to Do:

- Connect terminal
- Determine what device file to use
- Check `/etc/gettydefs` for appropriate entry
- Create entry in `/etc/inittab`
- Add entry to `/etc/ttytype`
- Rerun `init`
- Use `ps -ef` to verify `getty` is running on the port

Student Notes

Most of the time the device drivers required for the terminal will already be a part of `/etc/conf/dfi`. However, if you have edited your `dfile` you may want to verify that the driver you need is present.

If the `dfile` does not contain the driver you need to operate your terminal, you must add the driver t `/etc/conf/dfile` and remake the kernel.

Be sure of where on the interface you are plugging the terminal. The HP 9000 has both parallel and RS-232-C ports. These ports frequently appear identical. The built-in parallel interface should be label PARALLEL. Make sure you plug your cable into an RS-232-C port.

Adding a Terminal

To add a terminal to a MUX, perform these steps:

1. Connect the terminal to the desired port on the interface card.
2. Create the appropriate device file in the `/dev` directory. Communication ports (user terminals as well as modems) need to be identified by one or more device files, depending on the intended use of the port. Device file naming conventions vary, depending on the device's use. Terminal (`tty`) files are required for terminals.

   Terminal device files naming convention are:

   ```
   /dev/ttynn
   ```

   Example:

   ```
   # mknod /dev/tty01   c   1   0x205004   PortB
   ```

3. Check `/etc/gettydefs` and add a baud rate chain for the new terminal if necessary, or use an existing baud rate chain.
4. Edit `/etc/inittab` and add a `getty` for the new terminal.

   ```
   01:2:respawn:/etc/getty -h tty01 9600
   ```

5. Add an entry to the `/etc/ttytype` file

   The `/etc/ttytype` file is a database that contains the terminal type of the terminal associated with each port on the system. It is used by the `tset` and `login` commands. `/etc/ttytype` entries have this form:

   model_number location

   where:

   model_number is the product number of the terminal as defined in `/usr/lib/terminfo`

   location is the device file associated with the

   For example:

   ```
   2392      console          # System console
   2392       tty00            # Bill's terminal
   ```

6. Invoke `telinit q`
7. Invoke `ps -ef` to see if the `getty` is running.

Note A terminal `getty` can be temporarily turned off by changing the action from `respawn` to `off`. You will then need to kill the current `getty` process running. The `getty` will not

be respawned until you change the `off` back to `respawn`. Turning a `getty` off is usefu when troubleshooting a device file problem.

Removing a Terminal

If you want to remove a terminal from your system, follow these steps:

1. Determine the device file associated with the physical terminal. You can do this by logging into the terminal you wish to remove and issue the command `tty`.

```
# tty
/dev/tty04
```

2. Verify that no one is using the terminal by issuing a `ps -ef` command. You should find a `getty` running on the port.

```
# ps -ef |grep tty01
    root  1075     1  0  Oct 19  tty1p5   0:00 /etc/getty -h tty01 9600
    root  2830  2430  3 10:46:20 console  0:00 grep tty01
```

3. Edit `/etc/inittab` and delete the line for this terminal.
4. Kill the `getty` process for the terminal.

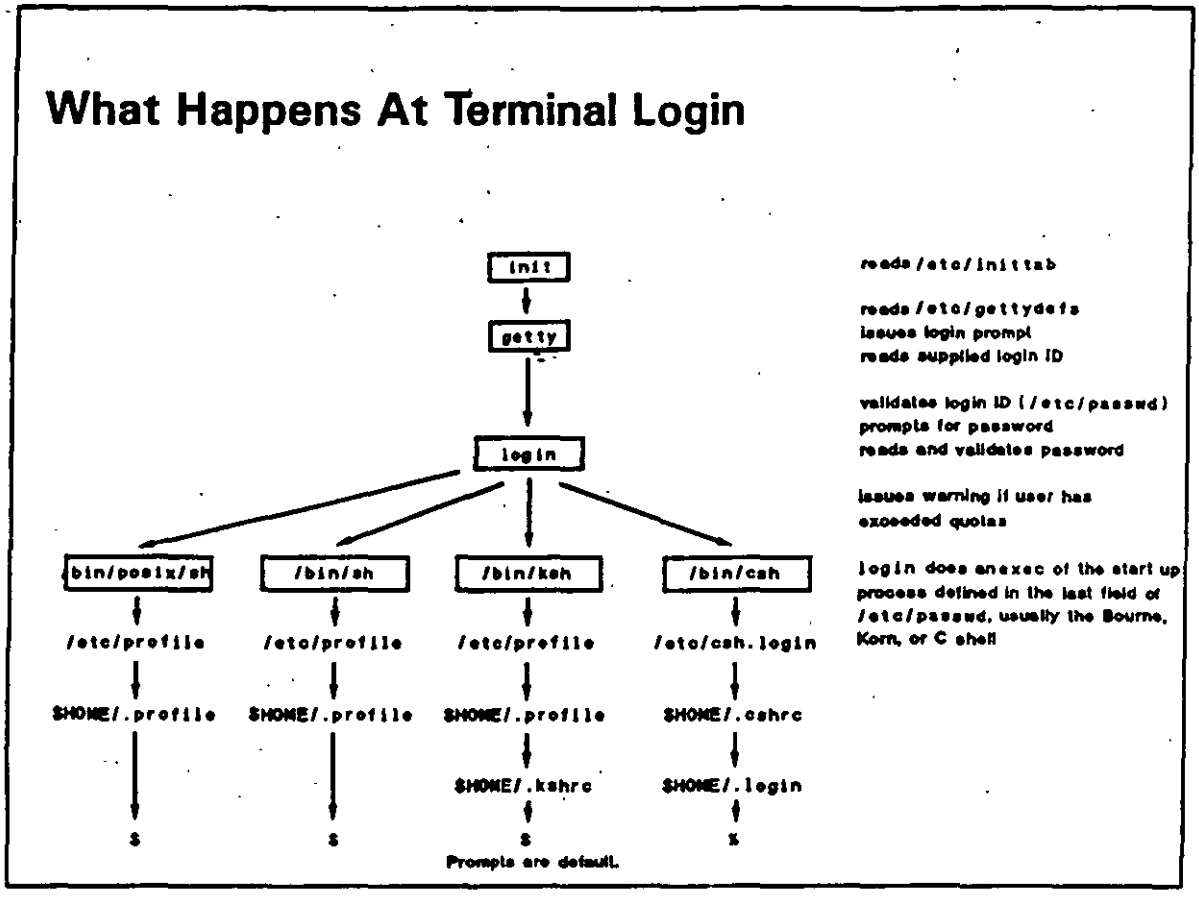
```
# kill -9 1075
```

5. Notify `init` that `/etc/inittab` has changed:

```
telinit q
```

6. Remove the device file associated with the terminal.

B-6. SLIDE: What Happens At Terminal Login



Student Notes

Allowing users to log into an HP-UX system is a three step procedure. The steps are:

`/etc/getty -> /bin/login ->` *command*

We have already discussed the operations of `getty`. Now we will discuss `login`. The following steps describe what `login` does.

1. `login` searches the `/etc/passwd` file for the username.
   a. if the username exits, `login` goes to step 2.
   b. if the username does not exist, `login`:
      i. Prompts the user for a password (`Password:`). This makes it difficult for an intruder to find a valid username.

ii. Displays the message `Invalid login`.

iii. If `/etc/securetty` is present, login security is in effect, meaning that only users with appropriate privileges are allowed to log in successfully on the ttys listed in this file.

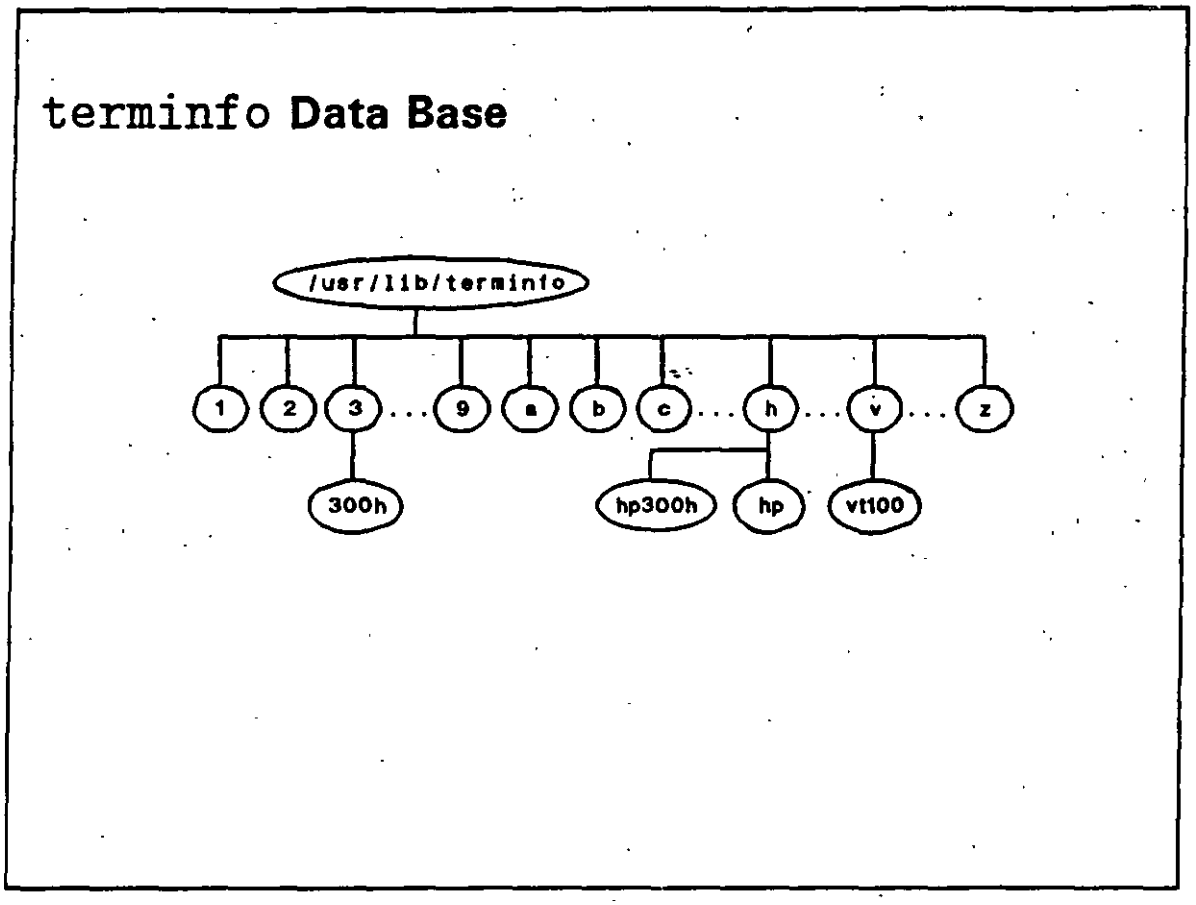
iv. If `/etc/btmp` is present, `login` logs all unsuccessful login attempts to this file.

v. If this is the user's third consecutive invalid login attempt, `login` exits; otherwise, `login` prompts the user (`login:`) for a username and repeats step 1.

2. `login` checks to see if the username's password field is set in `/etc/passwd`.
   a. If so, it prompts the user for a password (`Password:`) and goes to step 3.
   b. If not, then the user need not enter a password; go to step 4.
3. `login` compares the password to the username's encrypted password in `/etc/password`.
   a. If the password matches, `login` goes to step 4.
   b. If the password does not match, `login` displays the message `Invalid login`. If this is the user's third consecutive invalid login attempt, `login` terminates; otherwise, `login` prompts the user for a username (`login:`) and control passes back to step 1.
4. `login` sets the username's numeric user ID, group ID, and home directory from the corresponding fields in `/etc/passwd`. `login` also updates the `/etc/wtmp` file, which keeps track of valid logins.
5. `login` runs the `/etc/quota` command and warns the user if he or she has exceeded the quotas set by the system administrator.
6. `login` runs (via `exec` system call) whatever command is present in the command field of `/etc/passwd`. Typically, this field is set to the path name of the shell the user wishes to use; that is:
   a. `/bin/ksh` (Korn shell)
   b. `/bin/sh` (Bourne shell)
   c. `/bin/csh` (C shell)
   d. `/usr/bin/keysh` (Key shell)
   e. `/bin/posix/sh` (POSIX shell)

   If the command field is empty, `login` starts a Bourne shell by default.

B-8. SLIDE: terminfo Data Base



The contents of the environment variable TERM is a pointer to a file in the terminal capability database `terminfo`. Terminals are described in `terminfo` by giving a set of capabilities which they have and by describing how operations are performed.

For the sake of a faster access to the `terminfo` files, an additional level of directories was added. These directories are named after the first letter of the file names.

The `terminfo` database contains files describing HP terminals and many commonly used non-HP terminals. The set of files is updated by HP. Usually a system administrator need not modify these files. If you find you need to modify a terminfo file, translate it from an internal, compiled format into an open, source format with the `untic` command. The `tic` command compiles the modified source file and places the compiled file back into the `terminfo` database.