UNIX GENERAL Y CONCEPTOS REDES

ELABORADO Y DICTADO POR :

ING. HECTOR GIL TRIANA

Bucaramanga, Marzo de 2003

INDICE

1. FUNDAMENTOS DEL SISTEMA OPERACIONAL UNIX

1.1. EVOLUCION , VERSIONES.

UNIX es un Sistema Operativo multiusuario, multitarea, escrito en lenguaje de alto nivel ("C") en contraposición a un lenguaje ensamblador específico de alguna máquina. Esta característica (una de las más sobresalientes de este sistema) ha permitido que el UNIX sea "portado" (es decir, convertido) a una variedad de máquinas desde microcomputadores hasta supercomputadores.

Durante los años 60, los investigadores de los laboratorios Bell (de la AT&T) trabajaban en el desarrollo de un sistema operacional llamado Multics. Este proyecto fue cancelado en 1969, pero muchas de sus ideas influyeron posteriormente en el desarrollo del UNIX. De hecho, el nombre de UNIX se derivó del Multics. Sin embargo, mientras Multics era un sistema operacional extremadamente complejo, el UNIX es relativamente sencillo.

El desarrollo del UNIX fue iniciado por Ken Thompson (en Bell Labs) en un minicomputador PDP-7, por la misma época en que se canceló Multics (~**1970**). Sus objetivos eran:

- Crear un ambiente para el desarrollo de programas.
- Mantener un diseño simple.
- Diseñar un sistema operativo que fuera económico y corriera en el hardware disponible.

En **1973**, Dennis Ritchie reescribió el sistema en su nueva creación: el lenguaje de programación C. Como se explicó arriba, esta fue una de las mayores ventajas, ya que el sistema podría portarse a diversas plataformas de hardware con un mínimo de esfuerzo. La primera conversión se llevó a cabo en **1976**, a un Interdata 8/32.

Durante la década de los 70, el sistema empezó a ser aceptado internamente en la Bell. Hacia **1974**, Ken Thompson y Dennis Ritchie, publicaron su, ahora clásico, trabajo "The Unix Time-Sharing System", que fue acogido con gran interés. Por entonces, ellos ofrecían una copia de la cinta con los fuentes del sistema UNIX, **Versión 5**, prácticamente regalada.

Durante **1976** se inició la distribución de la **Versión 6**, especialmente a universidades en todo el mundo. Sobre esta versión inició su trabajo la Universidad de California en Berkeley.

La **versión 7** del sistema fue liberada en **1978**. Esta es muy importante ya que podría considerarse la raíz de todos los sistemas UNIX de hoy en día.

Dentro de los laboratorios Bell, el sistema UNIX se distribuyó posteriormente como **Release 3.0** y posteriormente, **4.0**. Iniciando la década de los 80, AT&T preparó una versión para distribución, que fue la 5.0, cambiado luego a **UNIX SYSTEM V**, para el cual creó un estándar conocido como "SVID" (System V Interface Definition).

Hacia finales de los 70 e inicios de los 80, el sistema UNIX había sido mejorado notablemente por estudiantes de la Universidad de California en Berkeley. Iniciando con la Versión 6, crearon la llamada 3 BSD (Berkeley Software Distribution) y luego la 4 BSD. Este sistema poseía una serie de mejoras tales como la *Shell* C, el editor visual (vi), lenguaje Pascal, soporte de redes, *sockets*, soporte de memoria virtual, etc.

Hoy en día podría pensarse que la mayoría de los sistema UNIX se derivan de una de estas dos fuentes (AT&T o BSD). Y de hecho muchos de los sistemas disponibles en el mercado de hoy en día combinan las mejores características de los dos sistemas.

Solaris, es el nombre con el que se conoce el sistema operativo de Sun Microsystems. Originalmente se llamó SunOS, pero posteriormente, debido a la presentación de UNIX Sistema V se desarrolló una nueva versión a la que se le llamó Solaris. Existen versiones de Solaris para Power PC, Intel y Sparc.

IRIS, es la versión de UNIX desarrollada por Silicon Graphics para sus estaciones y servidores basada en UNIX Sistema V version 4.

Linux es una implementación del sistema operativo UNIX (uno más de entre los numerosos clónicos del histórico Unix), pero con la originalidad de ser gratuito y a la vez muy potente, que sale muy bien parado (no pocas veces victorioso) al compararlo con las versiones comerciales para sistemas de mayor envergadura y

por tanto teóricamente superiores. [1]

Comenzó como proyecto personal del entonces estudiante Linus Torvalds, quien tomó como punto de partida otro viejo conocido, el Minix de Andy. S. Tanenbaum (profesor de sistemas operativos que creó su propio sistema operativo Unix en PCs XT para usarlo en su docencia). Actualmente Linus lo sigue desarrollando, pero a estas alturas el principal autor es la red Internet, desde donde una gigantesca familia de programadores y usuarios aportan diariamente su tiempo aumentando sus prestaciones y dando información y soporte técnico mutuo. La versión original y aun predominante comenzó para PCs compatibles (Intel 386 y superiores), existiendo también en desarrollo versiones para prácticamente todo tipo de plataformas: PowerPC, Sparc, Alpha, Mips, etc.

La primera fuente para conseguir el sistema Linux es la propia red Internet, y es donde estarán siempre las últimas versiones y las aplicaciones más actualizadas en muchos servidores de FTP anónimo. Otra vía muy frecuente, de interés para principiantes y para quienes no deseen o no puedan permitirse copiar tanta cantidad de información a través de la red, es mediante las versiones comercializadas en CDROM. Hay empresas que se dedican a elaborar CDROMs de bajo coste con recopilaciones de software, manuales, etc. El corazón del sistema es el mismo, aunque pueden tener externamente presentaciones y formas distintas de instalación. Hay revistas especializadas que también suelen incluir CDs con alguna versión de Linux. Hacia el final de este documento se relacionan diversas fuentes de Linux, tanto servidores públicos en la red como direcciones de empresas que lo comercializan.

Red Hat

Creada por Red Hat Software, en Connecticut, EE.UU. Una de sus ventajas es el atractivo sistema de instalación (en modo gráfico) y el cómodo mantenimiento de componentes de software, lo que facilita enormemente las tan frecuentes actualizaciones. Se puede obtener tanto gratuitamente en la red como adquiriendo el CDROM correspondiente. Otras empresas comercializan también sistemas basados en Red Hat, como Caldera Inc. y Pacific Hi-Tech. Aún poco conocida en España pero pujante, sobre todo para principiantes. Sus creadores estan en http://www.redhat.com. El mencionado sistema de gestión de componentes de software es obra suya, pero lo han ofrecido con carácter abierto y gratuito a los demás desarrolladores bajo la licencia de GNU, por lo que es

[1] Tomado de Articulo Spanish Linux Howto

previsible que en el futuro otros muchos asuman este sistema en sus propias distribuciones, lo que facilitará enormemente las actualizaciones.

1.2. OBJETIVOS DEL DISEÑO DEL UNIX.

- Sistema operacional y software de soporte sencillo, elegante y fácil de usar.

- Cada programa (herramienta) debía hacer sólo una cosa, pero muy bien hecha.

- Permitir que la salida de cualquier programa pueda ser usada como entrada a otro (posiblemente aún no definido).

- Hacer fácil la construcción de nuevas herramientas, suministrando para ello bibliotecas de procedimientos, bibliotecas de funciones y lenguajes de programación.

Todo esto con la ventaja adicional de ser altamente "portátil" por ser escrito en su mayoría en C.

Otras ventajas sobresalientes del UNIX son:

- Es un sistema de **propósito general**: permite ejecutar una amplia gama de trabajos y aplicaciones.

- Provee un ambiente **interactivo** que permite al usuario comunicarse directamente con el computador y recibir respuestas inmediatas a sus comandos.

- Ofrece un ambiente **multiusuario** que permite compartir recursos del computador entre usuarios sin sacrificar la productividad.

- Provee un ambiente **multiarea** que permite ejecutar más de un programa simultáneamente, a cada uno de los usuarios.

En la actualidad , la mayoría de versiones de unix, tienen interfaz gráfica muy amigable (KDE, CDE, etc) que facilita el trabajo. Sin embargo considero que es muy importante estar familiarizado con los comandos unix básicos, para poder entender la filosofía de este sistema operativo y fácilmente adaptarse a cualquiera de los ambientes de los diferentes proveedores.

1.3. ESTRUCTURA DEL SISTEMA OPERACIONAL UNIX.

El sistema operacional UNIX está conformado por una serie de programas que administran los diversos recursos del computador y que permiten el enlace entre el usuario y el computador.

Conceptualmente, el UNIX está basado en 4 componentes:

- El núcleo o kernel
- El interpretador de comandos o shell
- El sistema de archivos
- Las herramientas o programas utilitarios

1.3.1. El núcleo o kernel.

Es un programa que constituye el sistema operacional propiamente dicho. Coordina todo el funcionamiento interno del computador: maneja recursos (i.e. procesadores, memoria y dispositivos de entrada/salida) distribuyéndolos entre diferentes usuarios y diferentes tareas. Igualmente, controla los diversos procesos.

El kernel reside permanentemente en memoria principal y alguna parte está ejecutándose a todo momento. El código del kernel es guardado en un archivo ejecutable llamado unix. La mayor parte del sistema está escrito en C, y alguna pequeña parte en lenguaje assembler específico de la máquina.

La estrategia de diseño UNIX buscó que el kernel que fuera tan pequeño como fuera posible. Mucho del "sistema operacional" se "sacó" a programas compilados y ejecutados separadamente (que son las herramientas y utilidades).

El usuario no maneja directamente el kernel, sino que usa la "shell".

1.3.2. Shell.

Es un programa interpretador de comandos que actúa como puente entre el usuario y el kernel. Es iniciado por el kernel cuando un usuario entra al sistema.

Es, además, un lenguaje de programación que permite crear procedimientos automáticos (denominados "shell scripts").

Hay varios tipos de "shells":

- Bourne Shell. Es la estándar, despachada con todo sistema UNIX y viene del UNIX System V de AT&T.

- C Shell. Disponible inicialmente en los sistemas basados en BSD 4.x, aunque es corriente encontrarla prácticamente en cualquier sistema UNIX actual. Su sintaxis es muy similar al lenguaje de programación C.

- Korn Shell. Combina las mejores características de las dos anteriores, pero su disponibilidad es mucho más reducida.

- Bash Shell. El "Bourne Again shell" (Bash) fue creado para usarlo en el proyecto GNU. La intención fue que fuese el intérprete de comandos estandar en el sistema GNU. "Nació" oficialmente el domingo, 10 de enero de 1988. Brian Fox fué quien programó las primeras versiones de Bash y continuó actualizándolo hasta 1993. A principios de 1989, Chet Ramey empezó a ayudar a Brian y fué el responsable de muchos arreglos en el código y nuevas características. Muy común en todos los sistemas Linux y en los otros ambientes.

1.3.3. Sistema de archivos.

Es el aspecto más visible del sistema operativo, encargado de dar simplicidad al UNIX y proveer un método para organizar, recuperar y manejar la información. Posee una estructura de árbol, compuesto por un conjunto de archivos ordinarios, directorios y archivos especiales.

1.3.4. Herramientas y programas utilitarios.

Un programa es un conjunto de instrucciones para el computador. Los programas que pueden ser ejecutados por el computador directamente son llamados programas ejecutables o comandos.

Se podrían catalogar en los siguientes grupos:

- Editores de texto (vi, ex, sed). Con los ambientes gráficos soportados en la actualidad se han incorporado una serie de editores de texto, muy amigables para el usuario.
- Herramientas para manipulación de texto (grep, sed, awk, sort, etc.)
- Lenguajes de programación (C, FORTRAN, Pascal, COBOL, etc.). En algunos sistemas como Solaris, Iris, son productos que se venden por separado. Pero se pueden instalar las versiones GNU.
- Herramientas de soporte de desarrollo (make, configure, etc)
- Herramientas y comandos de administración del sistema . Algunos comandos son comunes a las diferentes versiones, pero aquí es donde más difieren los sistemas unix, pues cada fabricante incorpora sus herramientas de administración.

1.4. COMO ENTRAR AL SISTEMA (o "login") Y COMO TERMINAR.

1.4.1. Login.

Para que un usuario pueda establecer contacto con el sistema necesita :

- Una terminal en donde puede digitar sus peticiones al sistema, y a la cual el sistema pueda enviar sus respuestas. Antes se utilizaban terminales como VT100, ANSI, VT220, PT200. En la actualidad se utilizan PCs con emuladores de terminales (como hyperterminal de windows), conexiones remotas por telnet, o trabajo directo desde las diversas consolas o terminales virtuales del sistema.
- Un nombre de usuario que lo identifica como persona autorizado a ingresar al sistema. Más adelante se explicará como definir usuarios en el sistema.
- Opcionalmente, un password (o clave) que verifica su identidad. Generalmente debe ser al menos de 6 caracteres de longitud y contener

mínimo un carácter que no sea letra (máximo 8 caracteres). Esto se define en las políticas de manejo de cuentas de usuario de los diversos sistemas operativos.

Cuando la conexión es hecha el sistema escribe en la pantalla

login: ("prompt" de ingreso al sistema en una terminal de usuario).
ó

Console login : ("prompt" de ingreso al sistema en la consola de operación).

En este punto se debe teclear el nombre de usuario en minúsculas, seguido por **<return>**. Si el usuario ha definido una clave de acceso, el siguiente "prompt" es

password:

Se debe teclear la clave y terminar con **<return>**. Esta clave no será mostrada en la pantalla por razones de seguridad.

Si tanto el nombre usuario como su password asociado son correctos, el sistema mostrará los mensajes del día (si existen) y luego el "prompt" default del sistema que varía dependiendo de la shell .

Ejemplo:

```
Red Hat Linux release 7.2 (Enigma)
Kernel 2.4.7-10 on an i686
login: hector
Password:

hector@www hector]$
```

De este momento en adelante el sistema está listo para recibir comandos.

En los sistemas actuales, con interfaz gráfica, la pantalla de entrada viene acompañada de algunas opciones, como las veremos en los casos particulares a tratar.

1.4.2. Cambio de clave.

Es posible que usted desee incluir una clave con su nombre de acceso, o cambiar a clave que posiblemente le fue asignada. Pare ello, debe usar el comando **passwd**.

El diálogo sería el siguiente: (lo que el usuario digita se muestra en negrilla)

$**passwd**
passwd: Changing password for <su nombre de usuario>
Old password: **<debe digitar su clave actual. No verá eco>**
New password: **<digite la nueva clave deseada>**
Retype new password: **<digite la nueva clave otra vez, exactamente igual a la primera vez>**

Note que el sistema no muestra ninguna de las claves que usted digita durante la ejecución de este comando. Esta es la razón de que se solicite una confirmación.

Ejemplo:

```
[hector@www hector]$ passwd
Changing password for hector
(current) UNIX password:
New password:
Retype new password:
passwd: all authentication tokens updated successfully
[hector@www hector]$
```

El super usuario o usuario root, pueden cambiar el password a cualquier otro usuario. Para tal fin, simplemente digitan:

```
# passwd nombreuser
```

No se le preguntará la clave anterior, solo la nueva que debe digitar dos veces como verificación.

1.4.3. Logout o salida del sistema.

Una vez que ha terminado su trabajo en el computador, debe informarle que desea retirarse. Puede hacerse de dos formas:

<ctrl-d> al inicio de una línea ó, usando el comando

exit

En los interfaz gráficos existen otras formas para abandonar la sesión, desde el desktop.

1.5. LA LINEA DE COMANDOS DEL UNIX.

1.5.1. Sintaxis general.

Para hacer las peticiones comprensibles al UNIX se debe presentar cada orden siguiendo la "sintaxis de la línea de comandos". Esta sintaxis define el orden en el que deben digitarse los diversos componentes de un comando.

comando [opción(es)] [expresión] [nombre(s) de archivo(s)]

En donde,

- comando: es el nombre del programa que se desea ejecutar.

- opción(es): es una cadena de caracteres, usualmente precedida de un guión (p. ej **-la).** Una opción modifica la acción del comando o le da información adicional sobre la acción deseada. En todo caso el efecto de una opción depende totalmente de cada comando específico.

- expresión: es una cadena de caracteres que es usada como entrada al comando.

- nombre(s) de archivo(s): indica los archivos que serán manipulados por el comando. Si se omiten la mayoría de los comandos del UNIX suponen que deben leer del teclado.

La shell busca los comandos en los directorio definidos dentro de la variable PATH del usuario. Estos pueden ser el directorio de trabajo corriente (llamado "HOME Directory"); si no lo encuentra allí entonces en el directorio /bin (o /sbin) y si no, en el directorio /usr/bin (o /usr/sbin). Sin embargo la secuencia particular de búsqueda puede ser alterada redefiniendo la variable PATH, como se explica en el apartado sobre la shell.

Ejemplo de variable PATH para superusuario:

```
#echo $PATH
/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/sbin:/usr/sbin:/bin:/usr/bin:/usr/
X11R6/bin:/usr/local/bin:/root/bin
```

Ejemplo de variable PATH para un usuario normal:

```
# su - hector
$ echo $PATH
/usr/kerberos/bin:/bin:/usr/bin:/usr/local/bin:/usr/bin/X11:/usr/X11R6/bin:/home/hec
tor/bin
```

En esos directorios descritos dentro de la variable PATH , se hace la busca de los comandos.

En el ejemplo anterior se utilizó el comando " su – nombreuser". Esto se usa para cambiar de usuario (ID) y el "-" indica que se procesen todas las variables de ambiente definidas en el profile del nuevo usuario. Si se es el superusuario, no se pedirá clave, pero si se es un usuario normal y se desea cambiar a otro, si se pedirá la clave respectiva. Si se omite el nombre del usuario , es decir , solo se digita "su – " se entenderá como si se desea cambiar a la cuenta del superusuario y se pedirá la respectiva contraseña.

El comando más simple digitado en la línea de comandos UNIX, es una palabra aislada que por lo general llama un programa para ejecución.

1.5.2. Algunos ejemplos.

Comando **date**: es un programa que muestra la fecha y la hora.

```
$date
mar mar 25 14:25:49 COT 2003
```

Comando **who:** muestra los usuarios que están conectados al sistema indicando la línea usada y la fecha del inicio de sesión.

```
$ who
hector   tty1     Mar 25 14:26
webmaster tty2    Mar 25 14:27
```

```
root     pts/0    Mar 25 13:43
root     pts/1    Mar 25 13:43
root     pts/2    Mar 25 13:43
```

comando **who am i:** indica el nombre de usuario

```
$who am i
hgt      vt04        Nov 26 17:06
```

comando id: muestra el UID (número que identifica al usuario) del usuario actual , junto con el nombre y el GID (número que identifica a que grupo pertenece) y el nombre del grupo.

```
$ id
uid=500(hector) gid=500(hector) grupos=500(hector)
```

comando **echo:** escribe (hace eco) el argumento recibido, en la pantalla.

```
$echo esto es un ejemplo de echo
echo esto es un ejemplo de echo
```

1.5.3. Caracteres especiales.

Puesto que al digitar un comando es muy posible que se cometan errores, existen algunos caracteres que permiten realizar correcciones:

• Carácter de Borrado: generalmente definido como ^h (**<ctrl-h>**) ó backspace. Permite borrar el último carácter tecleado.
• Carácter de "Kill": generalmente definido como @, hace que la shell ignore línea corriente.

Si estos caracteres de borrado y "kill: no son de su agrado, puede modificarlos usando el comando **stty**.

$stty erase <carácter>

$stty kill <carácter>

Ejemplo:

```
# stty kill @
# wwho@ pwd
# /windows/hector/borrar/shell
```

En el ejemplo anterior, se redefinió el carácter de kill (anular una línea de comando) por el carácter @. Luego se procede a comprobar que al colocar este carácter frente a caracteres de la línea de comandos anula su efecto.

1.5.4. Ayuda en línea.

UNIX provee una ayuda que le dará información sobre los diversos comandos: **man**.

Comando **man**: este comando permite un rápido acceso a la documentación en línea, mostrando en la pantalla una imagen de las páginas del manual de referencia.

Por ejemplo, para consultar la página del manual del comando **ls**, teclee

$**man ls**

Y se despliega toda la información al respecto de ese comando.

Hay una opción muy importante del comando man, que puede ser de gran ayuda y es utilizarlo :

```
# man –k palabra-buscar
```

Busca en todas las páginas de los manuales y presenta aquellas que tengan palabra-buscar dentro de sus textos. Es importante cuando uno no recuerda algún comando especifico, pero sabe de que tema o que se desea hacer.

1.6. EL SISTEMA DE ARCHIVOS.

1.6.1. Introducción.

Un archivo es el principal vehículo para organizar y operar la información en un sistema. Es una colección de datos que reside en algún dispositivo.

Un sistema de archivos es una estructura de directorios completa. Incluye un directorio raíz y todos los demás subdirectorios "colgados" de éste.

Puede existir una correspondencia uno a uno entre sistemas de archivos y dispositivos o (más frecuentemente, cuando se tienen discos grandes) se puede tener más de un sistema de archivos por dispositivo.

El comando **df** puede usarse para hallar cuáles son los sistemas de archivos definidos en un sistema (y, de paso averiguar cuánto espacio queda disponible).

#df -k

Filesystem	1k-blocks	Used	Available	Use%	Mounted on
/dev/hdb2	4427436	3006688	1195840	72%	/
none	55264	0	55264	0%	/dev/shm
/dev/hdb1	15182608	13315440	1867168	88%	/windows

En todo caso, todo sistema de archivos, debe ser "montado" en el sistema, para que pueda ser accesado. Más adelante se explica cómo se realiza esta labor. Para ver la información en bloques de 1 K , dar **df -k** .

Es importante observar la columna de Use%, que indica el porcentaje de llenado de la partición. Para el caso de la partición / (raiz), no se debe dejar pasar de un 90%. Si se llega a llenar, el sistema puede caerse.

1.6.2. El árbol de archivos. Tipos de archivos.

Los archivos en UNIX están organizados en una estructura jerárquica de árbol con una relación padre-hijo.

Siempre se tiene un directorio raíz de donde salen todos los demás archivos y directorios. Se indica con /**.**

Existen además subdirectorios (que son como las ramas del árbol) y finalmente, archivos (las hojas el árbol).

Dentro de este árbol, se tienen los siguientes tipos de archivos:

- Archivos regulares u ordinarios, que pueden contener textos, datos, hojas electrónicas, programas ejecutables, etc.

- Directorios. Son realmente un tipo especial de archivo cuyo contenido es una lista de nombres de archivos, junto con información que permite hallar cada uno de esos archivos en el sistema.

- Archivos especiales. Son archivos asociados con dispositivos (orientados a carácter o a bloque) y se encuentran siembre en el directorio /**dev**.

Ahora, cuando se realiza el "login" el sistema coloca al usuario automáticamente en un directorio llamado "directorio de login" o "directorio HOME". El nombre de este directorio es generalmente el mismo de login, y puede estar bajo el directorio /home o /export/home**.** Dentro de su directorio, usted podrá crear archivos y directorios adicionales. También podrá borrarlos, moverlos y controlar el acceso a ellos, como se explicará mas adelante.

Para determinar en cualquier momento en qué directorio se encuentra, use el comando **pwd** ("print working directory") que muestra el nombre del directorio en el que está trabajando:

```
# pwd
/windows/hector/borrar/shell
```

Cómo saber el contenido de un directorio? Use el comando **ls** ("list"). Este comando lista los nombres de todos los archivos y subdirectorios en un directorio específico. Si no se especifica, lista los nombres de archivos y directorios del directorio corriente.

Por ejemplo,

```
# cd /home/hector
# ls -l
total 4
-rw-------   1 hector   hector       446 ene 28 09:41 mbox

# ls -la
```

```
total 48
drwxr-xr-x   4 hector   hector     4096   ene 28 09:41 .
drwxr-xr-x   4 root     root       4096   ene 28 09:26 ..
-rw-------   1 hector   hector      110   mar 25 14:28 .bash_history
-rw-r--r--   1 hector   hector       24   ene 28 09:26 .bash_logout
-rw-r--r--   1 hector   hector      191    ene 28 09:26 .bash_profile
-rw-r--r--   1 hector   hector      124   ene 28 09:26 .bashrc
-rw-r--r--   1 hector   hector      820   ene 28 09:26 .emacs
-rw-r--r--   1 hector   hector      118   ene 28 09:26 .gtkrc
drwx------   6 hector   hector     4096   ene 28 09:26 .kde
-rw-------   1 hector   hector      446   ene 28 09:41  mbox
-rw-r--r--   1 hector   hector     3511   ene 28 09:26 .screenrc
drwx------   2 hector   root       4096   ene 28 09:27 .xauth
```

Como se ve, este comando muestra la lista de archivos en orden alfabético, pero este listado puede ser producido de diferentes maneras. En el primer ls , solo se presentaba un elemento llamado mbox. En el segundo ls con la opción –a , se presentan más elementos.

Este comando tiene una gran cantidad de opciones. Las más frecuentemente utilizadas son:

-a Lista todas las entradas incluyendo aquellos archivos que comienzan con un punto (.). Sin esta opción ellos no son listados por el comando ls, y son llamados archivos invisibles o escondidos.

-l Lista el contenido del directorio en formato largo (como en el ejemplo anterior) dando información de tipo de archivo, permisos de acceso, números de enlaces, propietario, grupo, tamaño en bytes, y fecha de la última modificación para cada archivo.

-R Lista "recursivamente" los subdirectorios encontrados en el directorio solicitado.

En el listado en formato "largo" (opción -l), el primer carácter indica el tipo del archivo, así:

- archivo ordinario
d directorio
b archivo especial (dispositivo) de bloque
c archivo especial (dispositivo) de carácter
l link simbólico (sólo en BSD).

El número de "links" para un archivo se refiere al número de enlaces de ese archivo (explicado más adelante, comando ln). En el caso de un directorio esta columna se refiere al número de directorios inmediatamente bajo este directorio.

Note que en el ejemplo del listado de arriba, aparecen dos directorios con los nombres "." y "..". En todo directorio UNIX, existirán siempre estos dos directorios escondidos, ya que es la forma como el sistema de archivos mantiene su estructura de árbol. El directorio "." es un seudónimo del nombre del directorio corriente. El directorio ".." es un seudónimo del directorio padre del directorio corriente.

Otros ejemplos (ls recursivo):

```
# ls -lR /guillermo
/guillermo:
total 132
-rwxr-xr-x    1 guillerm root        3452 mar  5 09:21 9left.gif
-rwxr-xr-x    1 guillerm root        4165 mar  5 09:21 a-bandera.jpg
drwxr-xr-x    2 guillerm root        4096 mar 25 14:13 administracion
-rwxr-xr-x    1 guillerm root        5301 mar  5 09:21 a-escudo.gif
-rwxr-xr-x    1 guillerm root       49327 mar  5 09:21 arauca1.gif
-rwxr-xr-x    1 guillerm root       16830 mar  5 09:21 arauca.gif
-rwxr-xr-x    1 guillerm root        5203 mar  5 09:21 arauca.html
drwxr-xr-x    2 guillerm root        4096 mar 19 19:19 consultas
-rwxr-xr-x    1 guillerm root        6586 mar  5 09:21 esc2.gif
-rwxr-xr-x    1 guillerm root        7090 mar 19 19:32 index.html
-rwxr-xr-x    1 guillerm root        1481 mar  5 09:21 logo.gif

/guillermo/administracion:
total 52
-rwxr-xr-x    1 guillerm root        3452 mar  5 09:23 9left.gif
-r-xr-xr-x    1 guillerm guillerm      20 mar 25 14:09 clave
-rwxr-xr-x    1 guillerm root        6586 mar  5 09:23 esc2.gif
-r-xr-xr-x    1 guillerm root        2391 mar 25 14:13 index.htm
-rwxr-xr-x    1 guillerm root        9372 mar  5 09:23 logo.gif
```

```
-rwxr-xr-x   1 guillerm root        1921 mar  5 09:23 modifi12.php
-rwxr-xr-x   1 guillerm root        7911 mar 19 19:38 modifi1.htm
-rwxr-xr-x   1 guillerm root        2113 mar  5 09:23 modifi2f.php
-rwxr-xr-x   1 guillerm root        4092 mar  5 09:23 modifi2.htm

/guillermo/consultas:
total 84
-rwxr-xr-x   1 guillerm root        3452 mar  5 09:23 9left.gif
-rwxr-xr-x   1 guillerm root        7059 mar 19 19:12 consul1.htm
-rwxr-xr-x   1 guillerm root        3114 mar  5 09:23 consul1.php
-rwxr-xr-x   1 guillerm root        3726 mar  5 09:23 consul2copia.htm
-rwxr-xr-x   1 guillerm root        4175 mar  5 09:23 consul2f.php
-rwxr-xr-x   1 guillerm root        6381 mar 19 19:12 consul2.htm
-rwxr-xr-x   1 guillerm root        4796 mar  5 09:23 consul2.php
-rwxr-xr-x   1 guillerm root        6335 mar 19 19:12 consul3.htm
-rwxr-xr-x   1 guillerm root        3489 mar  5 09:23 consulta3.php
-rwxr-xr-x   1 guillerm root        3776 mar  5 09:23 indexcopia.htm
-rwxr-xr-x   1 guillerm root        3612 mar 23 14:29 index.htm
-rwxr-xr-x   1 guillerm root        9372 mar  5 09:23 logo.gif
-rw-r--r--   1 guillerm guillerm    5193 mar 19 19:19 modifi1.htm
```

1.6.3. Directorios y archivos de un sistema UNIX.

Esta organización puede variar un poco, entre los diferentes sistemas unix, pero básicamente los sistemas UNIX tiene los siguientes directorios básicos:

/ Este es el directorio raíz que es la base de todo el sistema de archivos. De allí se desprende toda la infraestructura de archivos y directorios de unix.

Ejm:

```
[root@www /]# cd /
[root@www /]# ls -l
total 232
drwxr-xr-x   2 root     root        4096       ene 23 14:39 bin
drwxr-xr-x   3 root     root        4096       ene 23 14:23 boot
drwxr-xr-x   2 root     root        4096       ene 28 09:21 capturas
```

```
drwxr-xr-x    9 carolina root        4096     mar 11 20:31 carolina
drwxr-xr-x    2 root     root        4096     ene 23 20:52 descargas
drwxr-xr-x   17 root     root       77824     mar 25 13:39 dev
drwxr-xr-x   72 root     root        8192     mar 25 14:27 etc
-rw-------    1 root     root          24 mar 22 18:20 filtros
drwxr-xr-x    4 gina     root        4096     mar 23 15:21 gina
drwxr-xr-x    5 guillerm root        4096     mar 18 23:04 guillermo
drwxr-xr-x    4 root     root        4096     ene 28 09:26 home
drwxr-xr-x    2 root     root        4096     un 21  2001 initrd
drwxr-xr-x    8 jhoana   root        4096     mar 11 20:31 jhoana
drwxr-xr-x    7 root     root        4096     ene 23 14:28 lib
drwxr-xr-x    2 root     root       16384     ene 23 13:27 lost+found
drwxr-xr-x    2 root     root        4096     ago 29  2001 misc
drwxr-xr-x    5 root     root        4096     ene 23 19:53 mnt
drwxr-xr-x    4 root     root        4096     ene 23 20:53 opt
dr-xr-xr-x   98 root     root           0     mar 25 08:38 proc
-rw-r--r--    1 root     root         109     ene 28 08:54 prueba
-rw-r--r--    1 root     root        5229     mar 19 15:01 pruebaftp
drwxr-x---   17 root     root        4096     mar 25 14:25 root
drwxr-xr-x    4 sayi     root        4096     mar 23 18:23 sayi
drwxr-xr-x    2 root     root        8192     mar 23 12:17 sbin
-rw-r--r--    1 root     root       11401     mar  6 14:15 sendmail.txt
drwxr-xr-x    3 root     root        4096     ene 23 14:30 tftpboot
drwxrwxrwt    9 root     root        4096     mar 25 14:16 tmp
drwxr-xr-x   16 root     root        4096     ene 23 14:06 usr
drwxr-xr-x   27 root     root        4096     ene 23 20:08 var
drwxr-xr-x    2 root     root        4096     ene 23 20:05 varios
drwxr-xr-x   30 root     root        8192     dic 31  1969 windows
```

/bin (o /sbin) Directorio contiene muchos de los programas de utilidad del sistema UNIX. Contiene la mayoría de comandos básicos.

Ejm:

```
[root@www /]# cd /bin
[root@www bin]# ls
arch          chown         domainname  gunzip    mkdir         ps          sleep   vi ash
        consolechars  echo        gzip      mknod         pwd         sort    view
ash.static    cp            ed          hostname  mktemp        red         stty
ypdomainname aumix-minimal  cpio          egrep       igawk     more          rm
```

```
 su      zcat awk          csh          ex          ipcalc    mount         rmdir      sync
zsh basename      cut          false      kill      mt          rpm       tar      zsh-4.0.2
bash          date          fgrep      ksh       mv          rvi       tcsh bash2          dd
       gawk       ln        netstat       rview      touch bsh          df          gawk-3.1.0
 loadkeys  nice          sed        true cat          dmesg          gettext     login
nisdomainname  setserial  umount chgrp        dnsdomainname  grep       ls
pgawk         sfxload    uname chmod         doexec         gtar       mail       ping
  sh         usleep
```

/dev Directorio que contiene todos los archivos de dispositivos. El contenido de este directorio puede ser extenso y solo se presentará una sección.

Ejm:

```
[root@www etc]# cd /dev
[root@www dev]# ls
adbmouse    hdg15      nst11a     ptyz5    sday8    sdcc2    sddh10  sds6
ttyD31    ttyu4
agpgart     hdg16      nst11l     ptyz6    sday9    sdcc3    sddh11  sds7       ttyd4
ttyU4
amigamouse  hdg2       nst11m     ptyz7    sdaz     sdcc4    sddh12  sds8       ttyD4
   ttyu5
amigamouse1  hdg3       nst12      ptyz8    sdaz1    sdcc5    sddh13  sds9       ttyd5
   ttyU5
apm_bios    hdg4       nst12a     ptyz9    sdaz10   sdcc6    sddh14  sdt        ttyD5
 ttyu6
ataraid     hdg5       nst12l     ptyza    sdaz11   sdcc7    sddh15  sdt1       ttyd6
                ttyU6
```

/etc Directorio que contiene muchos archivos con datos del sistema o con propósitos de mantenimiento, y configuración, por ejemplo:

/etc/cron Programa cronómetro para ejecución programada y desatendida de procedimientos.

/etc/default Valores por defecto para varios programas del sistema.

/etc/group Archivo con grupos de usuarios.

/etc/init Programa de inicialización del sistema, ancestro de todos los demás procesos del sistema

/etc/inittab Tabla de inicialización del sistema.

/etc/motd Mensaje del día, desplegado en las terminales durante el proceso de login.
/etc/passwd Archivo de usuarios

/lib Biblioteca de funciones del sistema, y librearias.

Ejm:

```
[root@www var]# cd /lib
[root@www lib]# ls
cpp                    libgcc_s.so.1          libnss_db.so.2.0.0     libpthread-0.9.so
i686                   liblber.so.2           libnss_dns-2.2.4.so    libpthread.so.0
iptables               liblber.so.2.0.6       libnss_dns.so.1        libpwdb.a
kbd                    libldap_r.so.2         libnss_dns.so.2        libpwdb.so
ld-2.2.4.so            libldap_r.so.2.0.6     libnss_files-2.2.4.so  libpwdb.so.0
ld-linux.so.2          libldap.so.2           libnss_files.so.1      libpwdb.so.0.61.1
libanl-2.2.4.so        libldap.so.2.0.6       libnss_files.so.2      libresolv-
2.2.4.so libanl.so.1               libm-2.2.4.so          libnss_hesiod-2.2.4.so
libresolv.so.2 libBrokenLocale-2.2.4.so    libmemusage.so
libnss_hesiod.so.2     librt-2.2.4.so libBrokenLocale.so.1    libm.so.6
 libnss_ldap-2.2.4.so   librt.so.1 libc-2.2.4.so              libNoVersion-2.2.4.so
libnss_ldap.so.2       libSegFault.so libcom_err.so.2         libNoVersion.so.1
   libnss_nis-2.2.4.so    libssl.so.0.9.6b libcom_err.so.2.0     libnsl-2.2.4.so
     libnss_nisplus-2.2.4.so  libssl.so.2 libcrypt-2.2.4.so       libnsl.so.1
libnss_nisplus.so.2    libss.so.2 libcrypto.so.0.9.6b       libnss1_compat-
2.2.4.so  libnss_nis.so.1        libss.so.2.0 libcrypto.so.2
libnss1_compat.so.1    libnss_nis.so.2        libtermcap.so.2
libcrypt.so.1          libnss1_dns-2.2.4.so   libpam_misc.so
libtermcap.so.2.0.8 libc.so.6                 libnss1_dns.so.1       libpam_misc.so.0
      libthread_db-1.0.so libdb-3.1.so          libnss1_files-2.2.4.so
libpam_misc.so.0.75    libthread_db.so.1 libdb-3.2.so
libnss1_files.so.1     libpam.so              libutil-2.2.4.so
libdb.so               libnss1_nis-2.2.4.so   libpam.so.0            libutil.so.1
libdl-2.2.4.so         libnss1_nis.so.1       libpam.so.0.75         libuuid.so.1
libdl.so.2             libnss_compat-2.2.4.so libpcprofile.so        libuuid.so.1.2
libe2p.so.2            libnss_compat.so.1     libpcre.la             modules
libe2p.so.2.3          libnss_compat.so.2     libpcre.so             security
libext2fs.so.2         libnss_db.so.1         libpcre.so.0
libext2fs.so.2.4       libnss_db.so.1.0.0     libpcre.so.0.0.1
```

```
libgcc_s-3.0.2-20010905.so.1  libnss_db.so.2        libproc.so.2.0.7
[root@www lib]#
```

/lost+found	Este directorio debe existir en todo sistema de archivos ya que el programa **fsck** deja aquí cualquier archivo hallado, sin estar referenciado en ningún directorio.
/mnt	"Punto de montaje" de sistemas de archivos removibles (diskettes, cdrom).
/tmp	Usado para almacenamiento temporal o de trabajo por muchos programas. Es borrado durante el proceso de boot. Tiene unos permisos especiales.
/usr	Contiene gran parte del sistema operativo, algunos archivos de administración del sistema, otros programas de utilidad, por ejemplo:
/usr/bin	Contiene más programas de utilidad del sistema
/usr/include	Archivos de encabezamiento estándar usados en programación en C
/usr/lib	Más bibliotecas de funciones.
/usr/lost+found	Como el del directorio raíz
/usr/spool	Manejo de colas (de impresión, de comunicaciones con otros computadores).
/usr/tmp	Almacenamiento temporal. Borrado durante el proceso de boot.

1.6.4. Pathnames y nombres de archivos.

Cada archivo en el sistema es identificado por un pathname o trayectoria, que describe una ruta desde la raíz a través de los directorios, hasta ese archivo particular.

Existen dos clases de pathnames:

- pathname absoluto: da el nombre de un archivo comenzando en el directorio raíz y descendiendo a través de una única secuencia de directorios hasta el archivo correspondiente. Por ejemplo,

  /windows/hector/borrar/shell/coms

  En este nombre, el primer "slash" representa el directorio raíz. Los siguientes "slashes" representan una ramificación en las estructura jerárquica de archivos: el nombre a la izquierda del slash es el directorio padre y el de la derecha, un hijo de ese padre. El nombre que sigue a continuación del

último slash (coms, en el ejemplo) se denomina un "nombre de archivo simple".

Note que todo pathname absoluto empezará por lo tanto con "slash".

• pathname relativo: da el nombre de un archivo comenzando en el directorio corriente y desciende a través de una única secuencia de directorios al archivo correspondiente.

Por ejemplo si el directorio corriente fuera /windows/hector un pathname relativo para el mismo archivo el ejemplo anterior podría ser

borrar/shell/coms

Un pathname relativo puede entonces empezar con:

• Un directorio o nombre de archivo simple.
• Un punto (.) para referirse al directorio corriente.
• Dos puntos (..) para referirse al directorio padre del directorio corriente.

Los nombres de archivos simples, deben seguir las siguientes reglas (algunas varían entre los diferentes sistemas Unix)

- Deben tener una longitud entre 1 y 255 caracteres.
- Pueden usarse todos los caracteres excepto el / ("slash")
- Debe tenerse en cuenta que UNIX diferencia las letras minúsculas de las mayúsculas. Así, ARC y arc son dos nombres diferentes.
- Debería evitar usar los caracteres +, -, y como el primer carácter en el nombre de un archivo.
- Debería evitar usar los caracteres
  ? @ # $ & * ( ) '[ ] / ! " < >
  dentro de un nombre de archivo, ya que ellos tienen significado especial para la "shell"

1.6.5. Cómo cambiar de directorio.

Cuando usted entra al sistema, usted es colocado en su directorio HOME; si desea trabajar en otros directorios use el comando **cd** cuyo formato general es

cd <path_del_nuevo_directorio>

El path puede darse en forma absoluta o relativa.

Si no se especifica ningún directorio, **cd** cambia al directorio HOME.

Ejm:

```
# cd /guillermo/administracion
```

1.6.6. Cómo examinar el contenido de archivos. Existen muchos comandos para mostrar el contenido de un archivo sobre la terminal. A continuación se verán algunos de ellos.

Comando cat.

$**cat [-vte] <archivo_1> <archivo_2> ...**

El contenido de los archivo es mostrado en la terminal (a menos que se redirija a otro archivo, como se explica más adelante). Algunas de las banderas de ejecución son:

-v Haga visibles los caracteres de control
-t Muestre los tabuladores
-e Muestre los caracteres de final de línea como un signo **$**

Ejm:

```
# cat .htaccess

AuthUserFile "/guillermo/administracion/clave"
AuthName "Digite Su Clave De Acceso"
AuthType Basic
require user admin
```

Comando more.

$more <archivo_1> <archivo_2> ...

Cada archivo es mostrado en la pantalla, pero si su longitud es mayor de un "pantallazo", el sistema se detiene y espera alguna tecla del usuario. Puede ser:

- <**barra espaciadora**>, para mostrar otro pantallazo.
- <**return**> para mostrar sólo una línea
- <**?**> para ver una pantalla de ayuda de las diversas teclas aceptables
- <**q**> para terminar aunque el archivo aún no se haya acabado.

Ejm:

```
# more /etc/httpd/conf/httpd.conf
#
# Based upon the NCSA server configuration files originally by Rob McCool.
#
# This is the main Apache server configuration file.  It contains the
# configuration directives that give the server its instructions.
# See <URL:http://www.apache.org/docs/> for detailed information about
# the directives.
#
# Do NOT simply read the instructions in here without understanding
# what they do.  They're here only as hints or reminders.  If you are unsure
# consult the online docs. You have been warned.
#
# After this file is processed, the server will look for and process
# /etc/httpd/conf/srm.conf and then /etc/httpd/conf/access.conf
# unless you have overridden these with ResourceConfig and/or
# AccessConfig directives here.
#
# The configuration directives are grouped into three basic sections:
--Más--(1%)
```

Comando tail.

$**tail <archivo_1> <archivo_2> ...**

Muestra por la unidad de salida estándar las 10 últimas líneas de cada archivo. Opcionalmente puede indicársele cuántas líneas del final se desean, con la opción -<n> ó a partir de cuál línea se desea, con la opción +<n>.

Ejm:

```
# tail -5  /etc/httpd/conf/httpd.conf

</VirtualHost>

</IfDefine>

#
```

Comando head.

$**head <archivo_1> <archivo_2> ...**
Muestra las 10 primeras líneas del archivo. Tiene la opción de variar este número indicando -<n>. Esta utilidad es propia de sistemas UNIX BSD y normalmente no se encuentra en UNIX System V.

Ejm:

```
# head -5  /etc/httpd/conf/httpd.conf
#
# Based upon the NCSA server configuration files originally by Rob McCool.
#
# This is the main Apache server configuration file.  It contains the
# configuration directives that give the server its instructions.
#
```

Comando pg.

Permite examinar el contenido de un archivo, página por página en una terminal. Es similar al comando **more** aunque un poco más poderoso, ya que permite devolverse en el archivo que se está examinando. Algunas versiones de unix no lo oportan.

1.6.7. Cómo determinar el tipo de un archivo.

A veces es útil saber cuál es el tipo de un archivo ordinario. Esto se logra mediante el comando **file**:

$**file <archivo_1> <archivo_2> ...**

Este programa intenta "adivinar" si el archivo dado es un programa ejecutable, un archivo de texto, un programa fuente, una librería, etc.

Ejm:

```
# file  /etc/httpd/conf/httpd.conf
/etc/httpd/conf/httpd.conf: ASCII English text
```

1.6.8. Como crear un directorio.

Es recomendable crear subdirectorios en el directorio HOME, de acuerdo con un esquema lógico. Para crear un directorio se usa el comando mkdir.

$**mkdir <nombre_del_directorio**>

El comando puede crear más de un directorio en una sola línea de comando. Los nombres de directorios, siguen las mismas reglas explicadas para nombres de archivos (al fin y al cabo, son un archivo). Si se desea crear toda una ruta se puede utilizar la opción –p.

Ejm:

```
# mkdir /pruebadir
# ls -ld /prueba
-rw-r--r--    1 root     root          109 ene 28 08:54 /prueba
# ls -ld /prueba*
-rw-r--r--    1 root     root          109 ene 28 08:54 /prueba
drwxr-xr-x    2 root     root         4096 mar 25 14:45 /pruebadir
-rw-r--r--    1 root     root         5229 mar 19 15:01 /pruebaftp

# mkdir -p /pruebap/otro
# ls -ld /prueba*
-rw-r--r--    1 root     root          109 ene 28 08:54 /prueba
drwxr-xr-x    2 root     root         4096 mar 25 14:45 /pruebadir
```

```
-rw-r--r--    1 root     root         5229 mar 19 15:01 /pruebaftp
drwxr-xr-x    3 root     root         4096 mar 25 14:46 /pruebap

# ls -lR /pruebap
/pruebap:
total 4
drwxr-xr-x    2 root     root         4096 mar 25 14:46 otro

/pruebap/otro:
total 0
#
```

1.6.9. Como copiar archivos.

Un usuario puede copiar archivos desde o hacia cualquier parte del sistema siempre y cuando tenga derechos e lectura sobre el archivo que quiere copiar (fuente) y de escritura sobre el directorio en el cual desea dejar la copia. Más adelante se explica el funcionamiento de los permisos.

La forma general del comando es

$**cp <archivo_fuente> <archivo_destino>**

Ejm:

$**cp test1.c test2.c**

En este caso se está creando una copia en el mismo directorio, con un nombre diferente

$**cp test1.c ../lab4/test1.c**

Aquí se está haciendo una copia del archivo en un directorio diferente, usando un pathname relativo, pero con el mismo nombre.

$**cp test1.c test2.c test3.c ../copias**

Este último ejemplo, muestra la forma de copiar una serie de archivos a otro directorio, dejándolos con el mismo nombre.

 # cp /etc/hosts /windows/hector/borrar

Se realiza una copia de un archivo a otro directorio cambiandole el nombre y utilizando rutas absolutas.

1.6.10. Como mover y/o cambiar de nombre.

Un usuario puede mover un archivo desde hasta cualquier parte del sistema siempre y cuando tenga permisos de lectura y ejecución en el directorio original del archivo y permiso de escritura en el directorio al que se moverá el archivo. El comando para este propósito es **mv**. A continuación se muestran varios ejemplos de uso:

$**mv test1.c test2.c**

Mover un archivo dentro del mismo directorio realmente es la forma de cambiarle de nombre.

$**mv test1.c ../lab4/test2.c**

Este ejemplo mueve el archivo a un directorio diferente cambiándole, además, el nombre.

$**mv test1.c test2.c test3.c ../lab4**

Aquí se han movido varios archivos a otro directorio, dejándoles el mismo nombre.

1.6.11. Como "encadenar" archivos.

A veces es conveniente tener una sola copia de un archivo con múltiples referencias a él. Esto es lo que se denomina en UNIX 'crear "links" a un archivo.

Se pueden crear "links" a un archivo en cualquier parte del sistema siempre y cuando el usuario tenga permiso de lectura del archivo que se desea encadenar y de escritura en el directorio en el que se desea crear el "link". El comando para realizar encadenamientos es **ln**. A continuación algunos ejemplos de su uso.

$**ln -s /usr/spool/lp sp**

Aquí se ha creado un link llamado sp , que apunta al directorio /usr/spool/lp .

$**ln -s test1.c ../lab4/test2.c**

En este ejemplo se ha creado un link en otro directorio y con otro nombre.

```
# cd /
# ln -s /etc/httpd/conf/httpd.conf /web
# ls -l /web
lrwxrwxrwx    1 root     root         26 mar 25 14:49 /web -> /etc/httpd/conf/httpd.conf

# tail -5 /web

</VirtualHost>


</IfDefine>

# wc -l /web
   1559 /web
# wc -c /web
  53571 /web
# wc -w /web
   7467 /web
#
```

Se crear un enlace llamado /web al archivo /etc/httpd/conf/httpd.conf. Luego cualquier operación que se ejecute sobre /web , es como si se realizara sobre el archivo verdadero, pues es un apuntador a el.

1.6.12. Cuántos caracteres, líneas, palabras.

El comando **wc** permite hallar el número de líneas, caracteres y/o palabras tiene un archivo.

$**wc [-wlc] <archivo> ...**

Las opciones son:

-w contar sólo el número de palabras
-l contar sólo el número de líneas

-c contar sólo el número de caracteres

El "default" son los tres.

1.6.13. Como imprimir archivos de texto.

La forma de imprimir en UNIX es solicitar este servicio al "spooler" de impresión. Esto se logra mediante el comando **lp o lpr.**

Por ejemplo,

$**lp test1 /etc/passwd**

colocará en la cola de impresión por defecto los dos archivos solicitados. Tan pronto esté disponible la impresora, serán escritos. El sistema responde con el nombre de la impresora en la cual el archivo será impreso y un número de identificación (ID). Este número podrá ser usado mas tarde para cancelar el listado si se desea o para averiguar si ya fue impreso. El comando tiene muchas opciones, algunas de las cuales son:

-d para especificar cuál impresora
-n para indicar el número de copias deseadas
-m para solicitar que el sistema informe vía correo la terminación de la impresión.
-w para indicar que se envíe un mensaje a la pantalla al terminar la impresión.

Algunas veces como usuario se necesita saber cual es el estado de la impresión que esta en progreso:

$**lpstat -t**

```
# lpstat -t
scheduler is running
system default destination: lexmark
printer lexmark unknown state. enabled since mar 25 14:55 2003. available
Printer: lexmark@www (spooling disabled)
 Queue: no printable jobs in queue
```

Existe también el comando **pr**, útil para "formatear" el contenido de un archivo. Si no se dan opciones, **pr** ajusta el texto a 66 líneas por página y añade dos líneas en

blanco al principio y al final de cada página. Además coloca en cada página una línea de identificación del archivo, junto con el número de la página.

1.6.14. Cómo borrar archivos y directorios.

Si no necesita mas un directorio usted puede borrarlo usando el comando **rmdir**, siempre y cuando tenga permisos de escritura en todos los directorios el pathname dado en el comando y el directorio esté completamente vacío.

$**rmdir <nombre_del_directorio>**

Para borrar archivos que ya no se necesitan más, use el comando

$**rm archivo(s)**

Ejm.

```
# rm comandos11
rm: remove `comandos11'? y
#
```

En este sistema que se capturó la salida del comando, hay una alias credo entre el comando “rm” y el “rm –i” que confirma antes de borrar. Si se desea suspender el alias, se puede digitar “unalias rm”.

Este comando tiene la opción -r, que borra recursivamente cualquier directorio o subdirectorio incluyendo los archivos dentro del directorio. Los archivos borrados con el comando rm no tienen forma de recuperarse, ha no ser desde una copia de seguridad.

Si los archivos se han borrado desde el interfaz gráfico , existe la posibilidad de recuperarlos desde la caneca (trash).

1.6.15. Permisos.

UNIX provee un medio para asociar un conjunto de permisos a cada archivo en su sistema de archivos.

Cada archivo tiene un propietario (inicialmente quien lo crea)

Solo el propietario de un archivo (o el super usuario), le puede cambiar los permisos, el dueño, o el grupo.

Respecto de permisos, hay tres clases de usuarios:

u: Propietario (**u**ser) - El usuario dueño del archivo, es decir el creador del archivo.
g: Grupo (**g**roup) - El grupo de usuarios al que pertenece el creador del archivo.
o: Resto (**o**thers) - Todos los otros usuarios del sistema (público).

Existen además tres tipos de permisos, cuyo significado es ligeramente diferente dependiendo de si el archivo es regular o es un directorio.

Para archivos regulares:

Read (r):	El contenido del archivo puede ser examinado
Write (w):	El contenido del archivo puede modificarse
Execute (x):	El archivo puede ejecutarse como comando

Para directorios:

Read (r):	Se puede ver el contenido del directorio
Write (w):	Se pueden crear y borrar archivos
Execute (x):	El directorio puede ser usado en un PATH.

Los tres tipos de usuario y los tres tipos de permisos dan un total de 9 posiciones de permisos en cada archivo o directorio. En un listado largo de un directorio (ls -l) los permisos se muestran como nueve caracteres, a continuación del tipo de archivo. Si en un archivo todos los tipos de usuarios tuvieran todos los permisos, se mostraría:

rwxrwxrwx

Los tres primeros caracteres corresponden al propietario, los siguientes tres al grupo y los últimos tres al público en general. Si algún permiso no existe para alguno de estos tres tipos de usuario, el sistema lo indica colocando un guión en el lugar correspondiente a ese permiso. Por ejemplo

rwxr-xr--

significaría que propietario tiene todos los permisos, el grupo no tiene permiso de escritura y el resto sólo tiene permiso de lectura.

Estos 9 caracteres se denominan colectivamente como el "modo del archivo".

UNIX tiene un comando el cual permite cambiar los permisos (i. e. el modo): **chmod**.

$**chmod <modo> <archivo(s)>**

El modo puede ser especificado en dos formas:

- Simbólicamente
- Número Octal

El modo puede ser especificado simbólicamente combinando una clase de usuario, un operador y un tipo de protección como se muestra en la tabla a continuación:

CLASE DE USUARIO	**OPERADOR**	**TIPO PROTECCIÓN**
u propietario	+ activa permiso	r Lectura
g grupo	- desactiva	w Escritura
o otros	= asigna	x Ejecución
a todos		

La interpretación de los permisos (rwx) , como ya se mencionó, tiene una interpretación si es archivo o directorio y los comandos que se pueden utilizar sobré el serían:

PERMISO	**ARCHIVO**	**DIRECTORIO**
R	Permite ver el contenido del archivo con comandos como cat, more, vi, etc.	Permite listar el contenido del directorio, con comandos como ls.
W	Permite editar el archivo y modificarlo. Por ejemplo editarlo y borrarle unas líneas.	Permite alterar la estructura del directorio, como crear subdirectorios, borrar archivos, cambiar nombres de archivos, etc.

X	Permite correr el archivo como un shell sin anteponer el nombre del shell respectivo.	Permite pasar en un pathname o situarse sobre un directorio con comandos como el cd.

Por ejemplo

$**chmod o-r texto**

quitará (-) a los otros (o) el permiso de lectura (r). Si texto tenía los permisos -rw-r--r-- , quedará con los permisos -rw-r-----

Para especificar el modo de protección como un número, se forma un número octal conformado por tres dígitos: el primero describe los permisos del propietario, el segundo los del grupo y el tercero los de los demás. Por ejemplo, si los permisos deseados son

rw- r-- ---

en binario sería

110 100 000 (1 si existe el permiso y 0 si no)
y en octal

6 4 0

El comando en este caso sería

$**chmod 640 texto**

Por otra parte, cuando usted crea un archivo o un directorio, el sistema automáticamente le da o le niega ciertos permisos. Estos permisos "default" pueden modificarse, usan el comando **umask** (que podría ser incluido en el archivo del perfil del usuario, .profile).

El formato de este comando es

$**umask [nnn]**

En donde nnn es un valor octal que determina cuales posiciones de permiso serán negadas.

Umask computa los permisos de acuerdo a la siguiente tabla:

ARCHIVOS		DIRECTORIOS	
Valor	Permisos	Valor	Permisos
0	rw-	0	rwx
1	rw-	1	rw-
2	r--	2	r-x
3	r--	3	r--
4	-w-	4	-wx
5	-w-	5	-w-
6	---	6	--x
7	---	7	---

Por ejemplo

$**umask 22**
producirá los siguientes permisos "default"

-rw-r--r-- archivo
drwxr-xr-x directorio

También relacionado con este tema, está el comando **chown**, que permite cambiar el propietario de un archivo. Sólo puede hacerlo el superusuario (administrador del sistema que siempre tiene todos los permisos en todos los archivos) o el propietario del archivo.

Por ejemplo

$**chown hgt prueba1 prueba2 prueba3**
cambiará al usuario hgt los archivos prueba1, prueba2 y prueba3.

1.7. EJERCICIOS DE REPASO PROPUESTOS

- Cambiar la clave de acceso al usuario con el que trabaja actualmente. Tratar de colocar una clave nula. Tratar de colocar una clave de dos caracteres.
- Supongamos que usted no recuerda el comando para instalar software. Buscar comandos relacionados con este tema.
- Indicar el porcentaje de llenado de la partición /. Que tamaño tiene libre en Mb?.
- Identificar a que grupo pertenece el usuario root?.
- En que directorio del sistema, encuentran archivo tipo “b” o “c” y que son?
- Manejo de rutas relativas y absolutas. Sin importar en que directorio estemos situados, cambiarse al directorio default , bajo /etc.
- Estando situados en /usr/bin, sin tener que pasar por /, cambiarse al directorio lib que esta al mismo nivel de bin.
- Tratar de crear desde comandos unix, un archivo cuyo nombre posea un espacio en blanco.
- Crear un link llamado /log , que apunte al directorio /var/log.
- Crear un directorio llamado /curso, dentro de /prueba. Si /prueba no existe, solo recurrir a un comando para crear los dos directorios.
- Usar el comando cat para crear un archivo de texto llamado /nuevo.
- Cuantas líneas tiene el archivo /etc/inittab ¿
- Mostrar en pantalla las 7 primeras líneas de ese mismo archivo.
- Mostrar en pantalla las 3 últimas líneas de ese mismo archivo.
- Verificar si hay alguna impresora definida con el sistema.
- Copiar el archivo /nuevo dentro de /prueba/curso , sin importar en que directorio estemos situados.
- Crear un archivo vacío llamado /prueba/curso/nulo. Si no conocen el comando, como lo buscan?
- Con que permisos se creo el archivo vacío? Porqué?
- Podría otro usuario de esa misma máquina borrarlo? Si o no y porque?
- Si la carpeta donde reside el archivo tiene permisos 755 y el archivo como tal tiene permisos 744, que daño le podrían causar al archivo.
- Creen un archivo estando como un usuario diferente de root. Ahora cambien el dueño del archivo a otro que resida en la máquina. Traten de cambiarle los permisos o de nuevo el dueño para que sea de nuevo de su propiedad. Que sucede?

1.8. LA SHELL.

Como se explicó al principio, la shell es un interpretador de comandos que actúa como puente entre el usuario y el kernel. Pero es en realidad mucho más que un simple interpretador. Tiene muchas capacidades que bien usadas pueden ayudar a hacer un uso más apropiado y eficaz del sistema.

En este capítulo del curso se explorarán algunas bases del la shell. En el capítulo 4 trataremos la parte de programación.

1.8.1. Archivos estándar y redireccionamiento.

La gran mayoría de los comandos del UNIX usan tres archivos definidos en forma estándar, para el manejo de sus operaciones de entrada/salida. Ellos son:

- stdin: archivo de entrada estándar, que por "default" es el teclado. Tiene el "descriptor de archivo" 0.
- stdout:archivo de salida estándar, que por "default" es la pantalla de la terminal. Tiene el "descriptor de archivo" 1.
- stderr: archivo de salida de errores de programas, por "default", la pantalla. Tiene el "descriptor de archivo" 2.

Ahora, la salida de un comando que escribe por *stdout* puede "redireccionarse", es decir hacer que sea enviada a un archivo diferente. Ello se hace usando el carácter **>** como en el siguiente ejemplo:

$**ls -l > list_dir**

Este comando listará el directorio corriente. El listado quedará grabado en el archivo list_dir.

El carácter de redireccionamiento se puede preceder de un número que indica el descriptor de archivo que debe redireccionarse (no es soportado en todos los shell)

$**comando 2>arch_errores**

En el ejemplo anterior, cualquier error quedará grabado en el archivo arch_errores.

El redireccionamiento puede hacerse a cualquier archivo (en el que el usuario tenga permisos de escritura), inclusive archivos especiales de dispositivos, por ejemplo:

$**ls -l >/dev/lp1**
enviaría la salida del comando a la impresora (si el puerto respectivo se llama lp1)

Si el archivo al que se redirecciona la salida ya existía, su contenido es borrado y reemplazado por la nueva salida. Si se desea conservar, añadiendo la nueva salida, se redirecciona usando **>>**. Por ejemplo:

$**ls -la /etc >> list_dir**
añadirá el listado del directorio /etc al archivo list_dir.

La información de entrada a un comando también puede redireccionarse, de manera que no se lea por el teclado si no de algún archivo de texto. Ello se logra con el carácter **<**.

Por ejemplo, suponga que existe un archivo "coms" con el siguiente contenido:

```
$cat coms
who
ps
```

Y que se ejecuta

```
$sh <coms
```

El comando **sh** es una invocación a la shell, que no leerá caracteres del teclado, sino del archivo coms. La salida será algo como

```
hcg       vt03        Jan 10 11:17
hcg       vt04        Jan 10 11:01
gic       ttyp0       Jan 10 11:17
  PID TTY      TIME COMMAND
  149 vt03     0:01 sh
  288 vt03     0:00 sh
  290 vt03     0:00 ps
```

que corresponde a la salida de los dos comandos who y ps.

En una misma línea de comandos es válido redireccionar tanto entrada como salida:

$**sh <coms >ejsalida**

1.8.2. "Pipelines" de comandos.

También es posible tomar la entrada estándar de un comando directamente de la salida estándar de otro comando. Esto es lo que se llama un "pipe" y se logra usando el carácter |.

Suponga, por ejemplo que desea saber cuantos archivos existen en su directorio corriente. Una forma de lograrlo, sería con la siguiente secuencia de comandos:

```
$ls >archtemp          # la salida del comando va archtemp
$wc -l tempfile        #ahora contamos el número de líneas
8
$rm tempfile           #borrar el archivo temporal.
```

Toda la secuencia de comandos anterior se puede reemplazar por:

```
$ls | wc -l
8
```

En este caso la salida del comando ls, se envía directamente como entrada al comando wc.

Otros ejemplos:

```
$ls -l /bin | more
```

Permitirá ver un directorio largo, con pausas cada pantallazo.

$**pr fuentes/prog1.c | lp**

El texto apropiadamente formateado, es enviado al spooler para impresión.

En una línea de comando es válido tener un "pipeline" con más de dos comandos.

1.8.3. Metacaracteres y generación de nombres.

Los metacaracteres son caracteres especiales que representan otros caracteres. Son algunas veces llamados "wildcards" o comodines. Son usados para generar nombres que concuerden con nombres de archivos o parte de nombres de archivos simplificando la tarea de especificar archivos o grupos de archivos como argumentos de los comandos.

En esta forma se simplifican tareas tales como borrar de un directorio todos los archivos que terminen en **.c**

Es importante entender que estos metacaracteres son usados por la shell para generar nombres de archivos basada en el significado del carácter, comparado contra el contenido del directorio especificado.

Los metacaracteres de la shell son:

*: Concuerda con cualquier string de caracteres incluyendo el string nulo. Usted puede usar el * para especificar un nombre de archivo total o parcial.

Ejemplos.
$**ls a***
Lista todos los archivos que comienzan con a.

$**more ***
Muestra por la pantalla el contenido de todos los archivos del directorio corriente.

?: Concuerda con un solo carácter de un nombre de archivo, pero puede ser usado más de una vez.

Ejemplos

$**ls capitulo?**
Lista todos aquellos archivos cuyo nombre comience con "capitulo" y siga con un solo carácter (capitulo1, capitulo5, capituloA, ...).

$**rm ???**
Borra todos los archivos cuyo nombre sea de (exactamente) tres caracteres.

[]: Cuando desea que el shell concuerde con uno cualquiera de varios posibles caracteres. Estos caracteres pueden aparecer en cualquier posición en el nombre del archivo.

Ejemplo: Si usted incluye [cmp] como parte de un patrón de nombre de archivo el shell mira por nombres de archivos que tengan la letra c, la letra m o la letra p.

$**ls [cmp]asa**
Listaría casa masa pasa , si esos archivos existen en mi directorio.

También es posible especificar rangos de letras o dígitos. separándolos con un guión. Por ejemplo, si se especifica capítulo[1-3] concordará con nombres como capitulo1, capitulo2 y capitulo3.

Un comando para borrar los directorios test1, test2, test3, test4, test5 y test8, sería:

$**rmdir test[1-58]**

Los diversos metacaracteres se pueden mezclar en sólo comando. El siguiente comando, por ejemplo, copiaría todos los archivos cuyo nombre empiece por una letra mayúscula y vaya seguido de al menos un carácter:

$**cp [A-Z]?* /usr/hgt/copias**

Cómo interpreta la shell los metacaracteres?

Para cada archivo que cumpla el criterio de selección (dado por los metacaracteres), la shell crea un argumento para el comando. Luego la shell arma el comando tal como si el usuario no hubiera usado metacaracteres, sino hubiera digitado los nombres completos. En esta forma, el comando que recibe los argumentos no "sabe" si el usuario usó o no metacaracteres y por lo tanto no es necesario que cada programa (comando) tenga el código necesario para manejar estos caracteres especiales. oda esta función está centralizada en la shell.

1.8.4. Variables de la shell.

Se pueden definir variables (nombres con algún contenido). El sistema tiene varias variables predefinidas:

- HOME: contiene el directorio "HOME" del usuario
- PATH: contiene una lista de pathnames usados por la shell para buscar (y ejecutar) un comando.
- LOGNAME: el nombre con el que el usuario hizo login
- PS1: "prompt" primario
- PS2: "prompt" secundario
- TERM: tipo de terminal en que el usuario está trabajando

Los valores de esta variables pueden hallarse usando el comando **env** (de environment, ambiente).

```
# env
PWD=/pruebap
HOSTNAME=www.sts.com
USER=root
MAIL=/var/spool/mail/root
BASH_ENV=/root/.bashrc
DISPLAY=:0
LOGNAME=root
SHELL=/bin/bash
USERNAME=root
HOSTTYPE=i386
QT_XFT=0
OSTYPE=linux-gnu
HOME=/root
TERM=xterm
PATH=/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin:/usr/local/bin:/root/bin
_=/usr/bin/env
OLDPWD=/windows/hector/borrar
```

También se puede usar el comando **echo**, así:

```
$echo $HOME
/root
```

Para asignar un valor a una variable (depende del tipo de shell que se trabaje), en sh, o ksh, o bash:

$**<nombre_de_variable>=<valor>**

Por ejemplo:

$**PS1=$LOGNAME**
cambiará el "prompt" estándar (signo $) a su nombre de usuario.
$**PATH=$PATH:/usr/wp/bin**
incluirá el directorio /usr/wp/bin en la variable PATH.

1.8.5. Caracteres de "escape".

Hay ciertos caracteres que tienen significado especial para la shell (metacaracteres, signo $, signo &, caracteres de redireccionamiento, etc). Si se requiere tratar estos caracteres literalmente (porque hacen parte del nombre de un archivo, por ejemplo) se debe informar de alguna forma a la shell que ignore el significado especial de tales caracteres. Ello se logra precediendo el carácter especial con el llamado carácter de escape, backslash (\).

Por ejemplo

```
$echo \*\* Esta es una prueba \*\*
** Esta es una prueba **
```

Si no se hubieran escapado los asteriscos, el comando echo habría reemplazado cada uno de ellos por todos los nombres de archivo en el directorio corriente.

Existen otros caracteres que prestan un servicio similar:

' (Comillas sencillas). Un string encerrado entre comillas sencillas no sufrirá absolutamente ningún tipo de expansión por parte de la shell: todos los caracteres especiales pierden su significado. Por ejemplo:

```
$echo 'Mi variable PATH contiene $PATH'
Mi variable PATH contiene $PATH
```

Note que la shell no reemplazó $PATH por el contenido de esta variable.

"" (Comillas normales). Similar al anterior, pero conservan su significado especial los caracteres \ ` y **$**. Por ejemplo:

```
$echo "Mi variable PATH contiene $PATH"
Mi variable PATH contiene /bin:/usr/bin:.:
```

`` (Comillas inversas). Esta es una característica muy poderosa. La shell ejecuta como comando el string encerrado entre estas comillas y lo sustituye por lo que el comando escriba por stdout. Ejemplo:

```
$echo "Numero de usuarios trabajando:`who|wc -l`."
Numero de usuarios trabajando: 5.
```

1.8.6. Creación y control de procesos.

Un "proceso" en UNIX es el término que se da a un programa que se encuentra en ejecución. Cada vez que usted da un comando, inicia un proceso. El sistema operacional, también puede iniciar procesos.

UNIX asigna a cada proceso un número de identificación único (llamado el PID: Process IDentification).

Cuando usted digita un comando, la shell normalmente crea (el término usado en UNIX es "fork") un proceso que es el que ejecuta el comando. En este caso la shell es un proceso **padre** y el comando, un proceso **hijo**. Mientras el proceso hijo está ejecutando la shell "se duerme" y cuando el hijo termina, muere y despierta al padre, que continua ejecutando.

Sin embargo, es posible indicarle a la shell que se desea que cree un proceso hijo, pero que no se duerma sino continúe ejecutando. Al hijo así creado se denomina un proceso "background" y la forma de hacerlo se ilustra en el siguiente ejemplo:

```
$proyecto &
979
```

El signo **&** indica a la shell que debe iniciar el proceso **proyecto** en el background, es decir, que debe continuar corriendo. La línea que escribe la shell indica el PID del proceso hijo creado.

Para averiguar qué procesos están ejecutándose en un momento dado, se tiene el comando **ps**:

```
$ps
  PID TTY      TIME COMMAND
  149 vt03     0:01 sh
 1001 vt03     0:00 ps
```

La información contenida en este informe es:

PID: Número de identificación del proceso
TTY: Terminal que controla el proceso
TIME: Número de segundos que el proceso ha estado ejecutando
COMMAND: El comando que inició el proceso.

El comando tiene varias opciones. Algunas de ellas:

-e Mostrar TODOS los procesos en ejecución en un momento dado
-l Listado largo, se muestra más información.

```
# ps -el
  F S   UID   PID  PPID  C PRI  NI ADDR    SZ WCHAN  TTY          TIME CMD
100 S     0     1     0  0  68   0   -   353 do_sel ?        00:00:04 init
040 S     0     2     1  0  69   0   -     0 contex ?        00:00:00 keventd
040 S     0     3     1  0  69   0   -     0 apm_ma ?        00:00:00 kapm-idled
040 S     0     4     0  0  79  19   -     0 ksofti ?        00:00:00 ksoftirqd_CPU0
040 S     0     5     0  0  69   0   -     0 kswapd ?        00:00:01 kswapd
040 S     0     6     0  0  69   0   -     0 krecla ?        00:00:00 kreclaimd
040 S     0     7     0  0  69   0   -     0 bdflus ?        00:00:00 bdflush
040 S     0     8     0  0  69   0   -     0 kupdat ?        00:00:00 kupdated
040 S     0     9     1  0  59 -20   -     0 md_thr ?        00:00:00 mdrecoveryd
040 S     0    13     1  0  69   0   -     0 end    ?        00:00:00 kjournald
040 S     0    88     1  0  69   0   -     0 end    ?        00:00:00 khubd
040 S     0   625     1  0  69   0   -     0 end    ?        00:00:00 eth0
140 S     0   702     1  0  69   0   -   368 do_sel ?        00:00:00 syslogd
140 S     0   707     1  0  69   0   -   496 do_sys ?        00:00:00 klogd
140 S    32   727     1  0  69   0   -   389 do_pol ?        00:00:00 portmap
140 S     0   839     1  0  68   0   -   349 do_sel ?        00:00:00 apmd
040 S    25   911     1  0  69   0   -  2880 rt_sig ?        00:00:00 named
040 S    25   912   911  0  69   0   -  2880 do_pol ?        00:00:00 named
140 S    25   914   912  0  69   0   -  2880 rt_sig ?        00:00:00 named
040 S    25   915   912  0  69   0   -  2880 nanosl ?        00:00:00 named
040 S    25   916   912  0  69   0   -  2880 do_sel ?        00:00:00 named
140 S     0   936     1  0  69   0   -   669 do_sel ?        00:00:00 sshd
140 S     0   956     1  0  68   0   -   566 do_sel ?        00:00:00 xinetd
140 S     0   984     1  0  69   0   -  1409 do_sel ?        00:00:00 sendmail
```

```
140 S    0  1003     1  0  69   0    -   360 nanosl ?        00:00:00 gpm
140 S    0  1026     1  0  69   0    - 12057 do_sel ?        00:00:01 httpd
000 S   26  1096     1  0  69   0    -  1655 do_sel ?        00:00:00 postmaster
140 S    0  1119     1  0  60   0    -   396 nanosl ?        00:00:00 crond
140 S   43  1189     1  0  69   0    -  1696 do_sel ?        00:00:02 xfs
140 S    0  1207     1  0  69   0    -   812 do_sel ?        00:00:00 smbd
140 S    0  1212     1  0  69   0    -   604 do_sel ?        00:00:00 nmbd
140 S    2  1248     1  0  69   0    -   361 nanosl ?        00:00:00 atd
140 S    0  1258     1  0  69   0    -  1205 do_sel ?        00:00:00 miniserv.pl
100 S    0  1262     1  0  69   0    -   589 wait4  tty1     00:00:02 login
100 S    0  1263     1  0  69   0    -   585 wait4  tty2     00:00:00 login
100 S    0  1264     1  0  69   0    -   346 read_c tty3     00:00:00 mingetty
100 S    0  1265     1  0  69   0    -   346 read_c tty4     00:00:00 mingetty
100 S    0  1266     1  0  69   0    -   346 read_c tty5     00:00:00 mingetty
100 S    0  1267     1  0  69   0    -   346 read_c tty6     00:00:00 mingetty
100 S    0  1268     1  0  68   0    -   625 rt_sig ?        00:00:00 kdm
100 S    0  1277  1268  0  78   0    -  9074 do_sel ?        00:00:33 X
140 S    0  1281  1268  0  69   0    -   832 wait4  ?        00:00:00 kdm
100 S    0  1309  1281  0  69   0    -   561 wait4  ?        00:00:00 startkde
140 S    0  1432     1  0  69   0    -  4071 do_sel ?        00:00:00 kdeinit
140 S    0  1435     1  0  69   0    -  4438 do_sel ?        00:00:00 kdeinit
140 S    0  1438     1  0  69   0    -  4522 do_sel ?        00:00:00 kdeinit
100 S    0  1439   956  0  69   0    -   609 do_sel ?        00:00:00 fam
100 S    0  1443     1  0  69   0    -  1368 do_sel ?        00:00:00 artsd
140 S    0  1460     1  0  69   0    -  4905 do_sel ?        00:00:00 kdeinit
140 S    0  1461     1  0  68   0    -  4129 do_sel ?        00:00:00 kdeinit
100 S    0  1462  1309  0  69   0    -  2887 do_sel ?        00:00:00 ksmserver
140 S    0  1463  1461  0  69   0    -  4497 do_sel ?        00:00:02 kdeinit
```

El contenido de algunas de las columnas es el siguiente:

F: Banderas asociadas con el proceso. Su valor puede ser uno de los siguientes, o sumas de varios de ellos.
00 Proceso terminado.
01 Proceso del sistema. Siempre en memoria principal
02 El proceso padre está haciendo "trace" de este proceso.
04 El proceso padre ha enviado una señal para detener el proceso.
08 El proceso se encuentra en este momento en memoria principal.
10 El proceso se encuentra en memoria principal, pero bloqueado hasta que se termine cierto evento.

S: Estado del proceso, que puede ser:

- O El proceso está corriendo en este momento.
- S El proceso está durmiendo, esperando a que se termine algún evento.
- R El proceso se encuentra listo para ser ejecutado, en cola.
- I "Idle". El proceso está en proceso de creación.
- Z Proceso en estado "Zombi": terminó y no tiene padre esperando a que termine.
- X El proceso está esperando a que haya más memoria disponible.

UID: Número de identificación del usuario propietario del proceso.
PID: Número de identificación del proceso.
PPID: Número de identificación del proceso padre de este proceso.
C: Utilización de procesador para control de este proceso (scheduling).
PRI: La prioridad del proceso (A mayor número, menor prioridad).
NI: Valor "Nice", usado en el cálculo de la prioridad (ver comando **nice** más adelante).
ADDR: Dirección de memoria física en la que se halla el proceso. Si muestra 0, significa que el proceso no está en memoria física en el momento de ejecutar el comando.
SZ: El tamaño del proceso (en páginas).
WCHAN: La dirección de un evento por el cual está esperando el proceso (que está durmiendo). Si es cero, el proceso está corriendo.
TTY: La terminal que controla el proceso (? si no hay terminal que lo controle).
TIME: Número de segundos que el proceso lleva ejecutando.
COMMAND: Línea de comando que arrancó el proceso

Si un proceso ya terminó pero su padre aún está esperando por él, es marcado como "<defunct>".

Los siguientes comandos, permiten control sobre los procesos:

Comando **kill [-numero_de_señal] lista_de_procesos**, envía una señal a los procesos especificados. El número de la señal, por "default" es 15, que significa terminar el proceso. Sin embargo, algunos procesos pueden ignorar tal señal. Si se especifica como señal **-9** el proceso no la puede ignorar y es la forma segura de matar un proceso.

Comando **nice**. Este comando se usa para ejecutar un comando con menor prioridad. A todo proceso se le asigna un valor "nice" que es, por default, 20. Entre mayor sea este valor, la prioridad calculada es menor. El comando **nice** permite incrementar este valor hasta un máximo de 39 (sumando al default de 20 el valor dado en el comando), con lo que el proceso se ejecutaría con mínima prioridad. Por ejemplo:

$**nice -10 compile &**
ejecutaría el comando "compile" con un valor nice de 30. (Si no se da un valor en el comando se toma un default de 10).

El superusuario, puede usar este comando con incrementos negativos, con lo cual el proceso se ejecutará con mayor prioridad. Por ejemplo:

#nice --8 proc_10
hará que el comando proc_10 se ejecute con un valor nice de 12.

1.8.7. EJERCICIOS DE REPASO PROPUESTOS.

- Ver el valor de la variable que almacena el prompt
- Almacenar el valor de la variable PATH en un archivo llamado rutas.
- Si los procesos identificados con un signo “?” iniciaron con el sistema, identificar los que no lo son.
- Crear con el comando cat un archivo llamado corre, que contenga la línea: ls –lR / por tres ocasiones y luego procesarlo desde una ventana de comandos. Inmediatamente abrir otra ventana y tratar de identificar el proceso respectivo corriendo en el sistema.
- Tratar de matarlo.
- Cuantos procesos corren actualmente en el sistema.

1.8.8. TEST DE AUTOEVALUACION

- Buscar el comando para creación de usuarios en Solaris
- Crear el siguiente directorio:

/copias lunes

- Crear la siguiente estructura de directorios (teniendo en cuenta que en la máquina existen los usuarios creados : lenin, unix, pruebas):

/home-lenin (dueño lenin, grupo other, permisos 754)
/home-unix (dueño unix , grupo other, permisos 755)
/home-pruebas (dueño pruebas, grupo other, permisos 764)
/todos (dueño root, grupo other) Todos los usuarios pueden escribir allí, pero solo pueden borrar lo que sea de cada uno de ellos.

- Explique brevemente que cosas y comandos pueden hacer los usuarios lenin, unix, pruebas y los demás del sistema en los directorios anteriormente creados.
- Copiar como root, a /home-lenin los archivos del directorio /etc que empiecen por n . Luego con un solo comando , cambiarles el propietario a unix a los archivos copiados.
- Copiar como root , a /home-pruebas los archivos del directorio /etc/init.d que tengan un número en alguna parte de su nombre. Luego con un solo comando , cambiarles el propietario a pruebas.
- Que cosas puede hacer el user lenin, sobre los archivos copiados?
- Que cosas puede hacer el user unix, sobre los archivos copiados?
- Mediante redireccionamiento hacer comandos para:
- (Antes de esto se deben crear mínimo 3 impresoras). Mostrar cuantas impresoras hay en el sistema (debe arrojar un número)
- Mostar los usuarios que estén corriendo el programa llamado: roo

El programa roo es un archivo de comandos unix que contiene:

```
echo  “escoja una opcion”
read opcion
echo $opcion
```

Debe tener permisos rwxr-xr-x y se puede situar inicialmente en la raiz (/).
Los usuarios lo deben correr : sh /root o bash /roo

- Si el programa roo esta en /todos, modificar la variable de PATH de root , para que busque también en ese directorio , programas o comandos del sistema y volver a lanzarlo.
- Matar el programa roo.

2. EDITORES EN UNIX

En la actualidad, los sistemas UNIX poseen gran variedad de editores de texto. Los interfaz gráficos proporcionan editores de muy fácil manejo. Los dos editores tradicionales de unix y que es bueno conocerlos, pues en algunas situaciones especiales (bootear máquina monousuario , o desde CD) son los únicos disponibles, son :

- El editor de línea **ed**.
- El editor de pantalla **vi** (que incluye un editor de línea mejorado llamado **ex).**

Al arrancar un editor, éste solicita al sistema un espacio de trabajo temporal llamado "buffer de edición" y cualquier información que usted entre cuando edita un archivo es almacenada en ese "buffer". Puesto que es temporal, cualquier texto que entre o cualquier cambio que haga, será también temporal. El "buffer" y su contenido sólo existe mientras usted esta editando el archivo; si desea salvarlo debe darle al editor el comando apropiado para que escriba ("salve") el "buffer" en un archivo permanente en disco.

2.1. EL EDITOR DE LINEA ED.

Es un editor muy rápido y es llamado de línea porque manipula el texto línea por línea. Fue el primer editor del UNIX y ha cambiado muy poco. Podría decirse que es el editor más universal del UNIX.

Aunque hoy en día, **ed** ha sido reemplazado por editores más poderosos (como el **vi**) y los de ambiente grafico , pero se justifica estudiarlo (al menos brevemente) por varias razones:

- Puede usarse en cualquier tipo de terminal, aún en una incorrectamente configurada.
- Típicamente, puede manejar archivos más grandes que otros editores.
- Tal vez lo más importante: por ser una de las primeras herramientas existentes en el mundo UNIX, muchos otros comandos han "copiado" su búsqueda de "expresiones regulares" (p. ej **grep, sed, awk, lex, ex**/**vi**), que es muy poderosa.

- En algunas situaciones especiales, es el único disponible.

En **ed**, por ser un editor de línea se habla de la "línea corriente" que es el texto que será generalmente modificado por los diversos comandos de edición. En los comandos se identifica esta línea por el carácter **"."** (punto).

Para arrancar una sesión de edición con **ed**, se invoca con el nombre del archivo que se desea editar, por ejemplo:

$**ed programa.c**

De aquí en adelante **ed** está listo para aceptar comandos. Por ejemplo, el comando **p**, mostrará en la pantalla la línea corriente. **20p** mostrará la línea 20 (que será ahora la corriente). **20,30p** mostrará las líneas 20 a 30 (y dejará la 30 como corriente).

Para añadir texto se usa el comando **a**. El texto que se introduzca de ahí en adelante, se agrega a continuación de la línea corriente. Se dejará de insertar texto al empezar una línea con el carácter **.**(punto).

Para grabar el buffer (o parte de él) se usa el comando **w**. Por ejemplo **1,30w parte1**, escribirá las líneas 1 a 30 en el archivo parte1. Si no se da rango de líneas, se entiende que es todo el buffer. Y si no se da nombre de archivo, se entiende que es el original.

El comando **q** termina la sesión de edición. Si hay modificaciones que aún no se han salvado, **ed** responderá con una **?** y no terminará. Si de todas formas se desea terminar se puede usar el comando **Q**.

Casi todos los comandos del **ed** pueden darse precedidos de un rango de líneas. Hay varias formas de dar tales rangos, como se ilustra en los siguientes ejemplos (con el comando **p**):

2p la segunda línea del archivo.
-1p la línea anterior a la corriente.
-5,+10p desde 5 líneas antes de la corriente, hasta 10 líneas después de la corriente.
/**programa**/**p** esta es una forma de direccionar "por contexto": se mostrará la primera línea que a partir de la corriente, contenga la palabra programa.

/programa/,/pascal/p mostrará desde la línea que contenga programa,hasta la línea que contenga pascal.
En los casos anteriores si en vez de / se usa **?** la búsqueda no se hará hacia adelante sino hacia atrás.
$p mostrará la última línea (**$** es sinónimo de "la última línea").
1,$p mostrará todo el archivo

El comando **=** muestra el número de la línea corriente

Para borrar líneas de texto, se usa el comando **d**. Por ejemplo **10d** borraría la línea número 10. **+10d** borra la décima línea a partir de la línea corriente. **d** borraría la línea corriente. **10,15d** borra la líneas 10 a 15. **20,/allá/d** borra de la línea 20 hasta la línea que contenga la palabra "allá".

El comando **i** se usa para insertar líneas antes de la línea corriente.

El comando **c** se usa para cambiar una línea (o rango de líneas) por otras que se digitan a continuación. Se termina con una línea que sólo contenga un punto

El comando **m** se usa para mover líneas. Ejemplos: **m50** mueve la línea corriente a después (enseguida) de la línea 50. **30m35** mueve la línea 30 a continuación de la línea 35. **/aquí/,40 m0** mueve desde la línea que contenga la palabra "aquí" hasta la línea 40, al principio del archivo.

El comando **s** se usa para sustituir texto. Por ejemplo:
1,$s/UNIX/LINUX/ cambiará la primera (atención: sólo la primera) ocurrencia de *UNIX* por *LINUX* en cada línea del archivo. Si se deseara cambiar todas las ocurrencias de la palabra buscada en cada línea, debe añadirse el modificador **g** al final:
1,$s/UNIX/LINUX/g. Si se desea observar cómo quedaron las líneas luego de la modificación, se puede añadir el modificador **p**.

El comando **r** se usa para leer otro archivo, insertándolo en el texto corriente. Por ejemplo **0r encabezado** insertará el contenido del archivo llamado "encabezado" al principio del buffer.

El comando **f** muestra el nombre del archivo que se está editando.

Existe además el comando "global" que sirve para dar como rango de direcciones de cualquier comando las líneas que contienen cierto patrón. Por ejemplo

g/editor/p imprimirá todas las líneas que contienen la palabra editor. Este comando realmente funciona en "dos pasadas": en la primera, se seleccionan todas las líneas que cumplen con el patrón dado (ver más adelante "expresiones regulares"). Después se ejecuta el comando dado sobre cada una de las líneas seleccionadas.

2.1.1. Expresiones regulares.

Todos los patrones de búsqueda del editor **ed** (y también de **grep, sed, awk, vi, ex)** pueden contener las así llamadas "expresiones regulares". Puesto que ellas son útiles en todos estos comandos, además del **ed**, hemos decidido incluir su estudio completo en estas notas.

Los siguientes caracteres tienen significado especial dentro de las expresiones regulares:

.	Punto
*	Asterisco
[]	Paréntesis cuadrados
\	Diagonal inversa
^	Acento circunflejo
$	Signo de pesos

Las expresiones regulares "complejas" son compuestas por expresiones regulares de un sólo carácter, que pueden ser:

- Un sólo carácter que no esté incluido dentro de los especiales (**.*[]\^$**). En este caso, el carácter sólo significa él mismo: **a** sólo coincidirá con **a**, por ejemplo.

- Un carácter especial "escapado": en forma similar a como ocurre en la shell, los caracteres especiales pierden su significado especial (y se interpretan textualmente) si se "escapan" precediéndolos de "backslash" (\). Así **\.** sólo coincidirá con el carácter **.**, *****, sólo coincidirá con el carácter *, etc.

- El carácter especial punto (**.**). Esta es una expresión regular de un sólo carácter que coincidirá con cualquier carácter (excepto el carácter de "fin de línea").

- El carácter especial paréntesis cuadrado (**[**), que indica que se inicia un conjunto de caracteres que termina cuando aparezca el otro paréntesis

cuadrado (**]**). El conjunto de caracteres que se encuentra de los paréntesis cuadrados, conforman una expresión regular de un sólo carácter que coincide con uno cualquiera de los caracteres del conjunto. Se puede usar un guión para indicar un rango, por ejemplo **[a-z]** coincidirá con cualquier letra minúscula.

Las expresiones regulares de un sólo carácter se pueden combinar para formar expresiones complejas, de la siguientes formas:

- Concatenación. Por ejemplo, la expresión regular **abc** sólo coincidirá con el string "abc". La expresión regular **a.c**, coincidirá con cualquier secuencia de tres caracteres que inicie con "a" t termine con "c".

- El operador asterisco (*****). Cuando una expresión regular de un sólo carácter va seguida de asterisco, coincide con 0 o más ocurrencias de esa expresión. Por ejemplo **12*3**, coincide con:
  13 (cero ocurrencias de la expresión regular **2**)
  123 (una ocurrencia)
  1223 (dos ocurrencias)
- 12223 (tres ocurrencias).

- El operador pesos (**$**). Cuando se coloca este operador al final de una expresión regular, tal expresión sólo coincidirá con el segmento final de una línea. (Si se coloca al principio o en medio de una expresión regular, el **$** coincide con "$"). Por ejemplo, la expresión regular **123$** coincidirá con la cadena de caracteres "123", sólo si ésta se encuentra al final de la línea. La expresión regular **$123** coincidirá con el string "$123" en cualquier parte de la línea.

- El operador acento circunflejo (**^**). Tiene dos significados: Cuando se usa como primer carácter de una expresión regular, tal expresión sólo coincidirá con el segmento inicial de una línea. Por ejemplo, la expresión regular **^Hola**, sólo coincidirá con el string "Hola" si éste se encuentra al principio de la línea.

- El otro significado se tiene cuando aparece como primer carácter de un conjunto de caracteres (i.e. entre paréntesis cuadrado). En tal caso, el conjunto de caracteres sólo coincide con caracteres que **no estén** en el conjunto dado. Por ejemplo, **[^a-z]** coincidirá con cualquier carácter que no sea una letra minúscula.

Donde hay duda sobre qué parte de un string coincide con una expresión regular, siempre se toma el criterio de la coincidencia más hacia la izquierda y más larga.

2.2 EL EDITOR VISUAL (O DE PANTALLA) VI.

El editor **vi** fue desarrollado en la Universidad de California en Berkeley. Es realmente muy poderoso. Posee una enorme cantidad de comandos que permiten desde sencillas labores de edición hasta operaciones muy complejas. En estos apuntes se incluirán los comandos usados con mayor frecuencia y unas cuantas de las características avanzadas.

Este editor permite ver el archivo que se esta editando (de ahí su nombre), una pantalla a la vez. El archivo es modificado posicionando el cursor en el lugar en donde se desea la modificación.

Debido a que **vi** usa abundantemente características específicas de la terminal (tales como posicionamiento de cursor, inserción y borrada de líneas, etc), es muy importante definir correctamente el tipo de terminal que se está usando. **vi** encuentra esta definición basándose en el contenido de la variable **TERM** (ver variables de la shell, en el apartado 1.8.4. de este mismo documento). Normalmente esta variable está correctamente definida en el archivo **.profile** (para ksh sh) del usuario, pero de no ser así, tendrá que definirla correctamente y exportarla.

El editor **vi** se encuentra en todo momento en uno de tres posibles modos:

- Modo de inserción de texto, en el cual obviamente se está insertando texto en el *buffer*.

- Modo de comandos, en el cual se reconocen y ejecutan una serie de comandos para manipular el contenido del archivo, dando la posibilidad de modificarlo. Este es el modo inicial.

- Modo "última línea", el cual se pueden dar comandos del editor **ex** o de la shell.

2.2.1. Cómo arrancar vi. Cómo terminar.

Para entrar al editor y crear (o editar, si ya existía) un archivo llamado "ejemplo" digite

$**vi ejemplo**

El editor traerá el archivo si ya existía o dejará la pantalla en blanco si se trata de uno nuevo. De todas formas el cursor quedará en la esquina superior izquierda de la pantalla. En este momento se encuentra en "modo comando".

Recuerde que se está editando una copia del archivo (en un buffer en memoria) y que éste no es alterado hasta tanto usted lo salve. La línea que contiene el cursor es la "línea corriente" y las líneas que contienen "tildes" (~) no forman parte del archivo, ellas indican líneas vacías.

Una vez se ha terminado una sesión de edición puede terminar (y retornar a la shell)con uno de los siguientes tres comandos (todos ellos actualizan el archivo editado, en el disco).

:x
:wq
ZZ

Si desea salir del editor olvidando todos los cambios realizados, digite

:q!

NOTA: es muy importante tener cuidado de diferenciar entre letras mayúsculas y minúsculas en los diversos comandos. **vi** posee una gran cantidad de comandos y muchos de ellos usan letras mayúsculas y minúsculas con sentido totalmente diferente (por ejemplo **d** y **D** son dos comandos diferentes).

2.2.2. Movimiento del cursor.

Si la pantalla tiene teclas de flechas, éstas son la manera más fácil de mover el cursor, en el sentido correspondiente (el **vi** tiene que estar en modo comando para que realicen la función esperada. En caso de duda, una buena forma de garantizar que el editor está en modo comando, es oprimir dos veces la tecla <**esc**>).

Además de las teclas de flechas, las siguientes letras también mueven el cursor (Observe que estas teclas se escogieron de acuerdo a sus posiciones en el teclado, de una forma mnemotécnica).

h mueve el cursor un carácter a la izquierda (como flecha hacia la izquierda).

j mueve el cursor una línea abajo (como flecha hacia abajo)

k mueve el cursor una línea arriba (como flecha hacia arriba).

l mueve el cursor un carácter a la derecha(como flecha hacia la derecha).

Note que el cursor no puede moverse a lugares "vacíos", es decir en donde no hay texto.

Otros comandos de movimiento de cursor, son los siguientes:

w (word) mueve el cursor al principio de la siguiente palabra
b (back) mueve el cursor hacia atrás una palabra
e (end) mueve el cursor hasta el próximo final de palabra

0 ó **^** mueve el cursor al comienzo de la línea
$ mueve el cursor al final de la línea

<**ctrl**>-**d** (down) mueve el cursor hacia abajo medio "pantallazo"
<**ctrl**>-**u** (up) mueve el cursor hacia arriba medio "pantallazo"

<**ctrl**>-**f** (forward) mueve el cursor hacia abajo un "pantallazo"

<**ctrl**>-**b** (backward) mueve el cursor hacia arriba un "pantallazo"

G mueve el cursor a la última línea del archivo
<n>**G** mueve el curso a la línea n, en donde n es un número

2.2.3. Comandos para agregar texto.

Para agregar algún texto en un archivo, use el comando **i** (insert), que cambia del modo comando, al modo inserción de texto. Todo lo que usted digite de aquí en

adelante, será insertado en el *buffer*, antes del sitio en donde se encontraba el cursor.

Para salir del modo inserción de texto y volver entrar al modo comando (es decir, para terminar la inserción de texto) presione la tecla <**esc**>.

Otros comandos que también pasan el **vi** al modo de inserción de texto, son los siguientes (todos ellos, por supuesto, terminan con <**esc**>):

a (append) añade texto a continuación de la posición corriente del cursor.

I posiciona el cursor al principio de la línea de la línea y entra a modo inserción (equivalente a **0i**).

A posiciona el cursor al final de la línea de la línea y entra a modo inserción (equivalente a **$a**).

o (open) abre una línea a continuación de la línea corriente (sin importar en donde esté el cursor) y entra en modo inserción)

O abre una línea arriba de la línea corriente (sin importar en donde esté el cursor) y entra en modo inserción)

NOTA: siempre que se está en modo inserción puede usarse la tecla de backspace, para corregir errores de digitación.

2.2.4. Comandos para borrar texto y para recuperar texto.

Los siguientes comandos se usan para borrar texto:

x borra el carácter sobre el cursor
X borra el carácter a la izquierda del cursor
dw borra desde el cursor hasta el final de la palabra en la que se encuentra el cursor
db borra desde el cursor hasta el principio de la palabra en la que se encuentra el cursor
d$ borra desde el cursor hasta el final de la línea (**D** hace lo mismo)
d0 (ó **d^**) borra desde el cursor hasta el comienzo de la línea
dd borra la línea sobre la que se encuentra el cursor.

(De todas las anteriores las que más se usan son x, dw y dd).

Muchas veces se cometen errores (se borra algo que no se debía borrar o se modifica equivocadamente algo). Con **vi** es muy fácil "arrepentirse" con el comando **u** (undo). Al darlo inmediatamente, **vi** "deshace" el último comando de borrada o de modificación dado. El comando **U**, por otra parte, deshace todos los comandos de modificación de una línea y la deja tal como estaba cuando el cursor se posicionó inicialmente en tal línea. (Pero si el cursor se posicionó en una línea distinta, ya no es posible recuperarla).

Casi todos los comandos del **vi** pueden precederse de un número, que es interpretado como un factor de repetición. Por ejemplo

5dd significa borrar 5 líneas.

Cuando se realiza cualquier borrado, las líneas que se borraron se copian a un "buffer" temporal de donde poden recuperarse con el comando **p** (paste). Al recibir este comando, **vi** inserta el texto más recientemente borrado, a partir de la línea en la que se encuentra el cursor.

Note que esta sería una manera fácil de mover líneas de un sitio a otro: primero se borrarían con <**n**>**dd**. Luego se colocaría el cursor en el sitio al que se desean mover y se daría el comando **p**.

2.2.5. Manejo de archivos.

Mientras se está trabajando en la edición de un archivo, es conveniente salvarlo de cuando en cuando (no es mala idea hacerlo cada 10 minutos). Para ello se usa el comando **:w** (write) que graba el archivo (con el mismo nombre) y permite continuar editando. (Note los dos puntos antes de la w. Al darlos, **vi** pasa al "modo última línea" en el cuál entiende básicamente un conjunto de comandos iguales a los del **ed**).

Este comando también puede usarse con un nombre de archivo para salvar con nombre diferente. Por ejemplo **:w copia_1** salvará con el nombre copia_1 el contenido del *buffer*.

Algunas veces se cometen tantos errores de edición que resulta preferible iniciar nuevamente. Es entonces conveniente terminar la sesión con **:q** (quit). Sin

embargo, **vi** no permite salirse sin salvar, si hay cambios. Para ello debe darse el comando seguido de **!** que es la forma de confirmarle que de todas formas se desea terminar (**:q!**).

Por otra parte, no es necesario salir del editor para iniciar la edición de otro. Basta con salvar el trabajo (**:w**) y luego dar el comando **:e <nombre_archivo>**. También se puede usar **:e!** si se desea editar otro archivo sin salvar el corriente. El comando **:e#** permite volver a editar el archivo anterior.

vi también provee el comando **:r** (read) para leer un archivo,insertándolo en la posición corriente del cursor.

2.2.6. Comandos para modificar texto.

r (replace) reemplaza el carácter que se halla sobre el cursor, por el próximo carácter digitado.

R entra en "modo reemplazo". En este modo todos los caracteres digitados reemplazan a los anteriores, hasta que se digite <**esc**>.

Dentro de este grupo, también se encuentran los comandos de "cambio de texto":

cw (change word) cambia la palabra sobre la que se encuentra el cursor por lo que el usuario digite. Termina con <**esc**>. (**vi** coloca un signo $ para indicar cuál será el sitio hasta donde se efectuará el cambio).

c$ cambia desde el cursor hasta el final de la línea. Es equivalente a **C**.

c0 cambia desde el principio de la línea hasta la posición del cursor.

2.2.7. Búsqueda y reemplazo de patrones.

Usted puede buscar un patrón de caracteres digitando un slash (/). **vi** entra en el modo "última línea", colocando el cursor en la última línea. De este momento en adelante puede digitarse el patrón que está buscando. Tal patrón es una expresión regular, que sigue las reglas explicadas en el apartado 2.1.1. El cursor se posicionará en la primera palabra que coincida con la expresión dada, a partir de la línea corriente.

Si se desea buscar la siguiente ocurrencia presione **n** (next). **N** busca la ocurrencia anterior.

Si se desea buscar del cursor hacia atrás se usa el carácter **?** en lugar del /.

Para reemplazar todas (o algunas) ocurrencia de un patrón determinado por otro, se usan comandos del editor **ex**, que es muy similar al **ed** descrito anteriormente. Cuando se oprime **: vi** también pasa al "modo última línea" en el cual se puede dar cualquier comando del **ed**.

Por ejemplo para cambiar todas las ocurrencias de la palabra "texto" por "documento" en el buffer, se daría la instrucción

:1,$s/texto/documento/g

(Vea el comando **s** en al apartado 2.1, página 59).

Este comando significa "desde la primera línea (1), hasta el final del archivo ($) encuentre "texto" y reemplácelo por "documento" en cualquier lugar en que este se encuentre en cada línea (g).

Otra forma puede ser con el comando global (vea página 59):

:g/texto/s//nuevo_texto/g

Que dice "seleccione cada línea del archivo que contenga la palabra "texto" (g /texto/) y en cada una de esas líneas ejecute el comando **s**ustituya la palabra "texto" por el "nuevo_texto" (s//nuevo_texto/) todas las veces que ocurra en la línea (g).

En el comando anterior se puede añadir al final el modificador **c** (confirm) con el que **ex** pedirá confirmación antes de realizar cada cambio. Para ello mostrará la línea en la que encontró el patrón y esperará una respuesta. Si tal respuesta es **y** el cambio se realizará. Cualquier otra respuesta indica que nos debe realizar el reemplazo.

Por supuesto en estos comandos de reemplazo, el patrón buscado es una expresión regular (el patrón de reemplazo es un "string" normal).

2.2.8. Marcadores.

Para moverse con agilidad en un texto extenso, es conveniente colocar marcadores que permitan saltar rápidamente a ellos. Con este propósito se usa el comando **m** (marker) seguido de una cualquiera de las 26 letras del alfabeto inglés. Es decir, se tiene 26 posibles marcadores.

El marcador quedará colocado en la línea en la que se encuentra el cursor.

Si se desea que el cursor se localice en un marcador determinado, se usa **'**. Por ejemplo **'a** significará "posicionar el cursor en el marcador **a**".

Todos los comandos del **ed** que usan números de línea para describir un rango, son aceptados por **ex**. Pero, adicionalmente, pueden usarse marcadores en vez de números de líneas.

Por ejemplo, para borrar un grupo de líneas cualesquiera, se podría colocar un marcador (por ejemplo el **a**) en la primera línea del grupo a borrar y otro marcador (por ejemplo el **b**) en la última. Enseguida se daría el comando **:'a,'bd** para borrar el grupo de líneas.

Otra forma de lograr este mismo propósito es colocar un marcador al principio del bloque que se desea borrar (por ejemplo **ma**), luego mover el cursor al final del bloque a borrar y dar el comando **d'a** (que significa "borrar hasta el marcador a").

2.2.9. Comandos para cortar y pegar texto.

El comando **y** (yank) permite copiar líneas a un "buffer" de donde luego puede ser recuperadas con el comando **p**. Este comando tiene varias formas (note que son totalmente similares a las del comando **d**):

yw copia desde el cursor hasta el final de la palabra en la
que se encuentra el cursor
yb copia desde el cursor hasta el principio de la palabra en
la que se encuentra el cursor
y$ copia desde el cursor hasta el final de la línea
y0 (ó **y^**) copia desde el cursor hasta el comienzo de la línea

yy copia la línea sobre la que se encuentra el cursor.

También pueden usarse factores de repetición. Por ejemplo **5yy** significará copiar 5 líneas.

Lo mismo que en el caso del comando **d**, también pueden usarse marcadores para copiar bloques.

2.2.10. Buffers.

Anteriormente se ha explicado cómo los comandos **d** y **y** copian líneas a un buffer. **vi** usa normalmente un buffer "default". Pero el usuario puede seleccionar uno de 26 posibles buffers, precediendo el comando con **"<letra>** en donde "letra" es una de las 26 del alfabeto.

Por ejemplo, el comando **"a10yy** copiará 10 líneas (a partir del cursor) al buffer **a** y allí se quedarán hasta que tal buffer sea utilizado nuevamente por el usuario.

Para recuperar líneas de un buffer específico, se precede el comando **p** de **"<letra>**. Por ejemplo **"ap** pegará la líneas que se habían copiado al buffer **a**, a continuación del cursor. El hecho de usar en este forma el buffer, no hace que se pierda su contenido. Es decir, puede volverse a pegar en otra parte.

Una aplicación de lo anterior, es copiar un bloque de texto de un archivo a otro. Suponga, por ejemplo, que se requiere insertar una copia de las líneas 50 a 80 del archivo "capítulo.1", al final del archivo "apéndice". La siguiente secuencia de comandos lograría este propósito:

$**vi capítulo.1**	para arrancar el editor
50G	coloca el cursor en la línea 50
"q30yy	copia 30 líneas en el *buffer* **q**
:e apéndice	cambia a editar el archivo "apéndice"
G	se posiciona al final del archivo
"qp	recupera las líneas del *buffer* **q**

2.2.11. Otros comandos.

Se incluyen aquí algunos otros comandos del editor, que aún no han sido mencionados en estas notas.

. Un punto digitado en modo comando, repite el último cambio

<**ctrl**>**g** (equivalente a **:f**) Informa el nombre del archivo y el número de la línea sobre la que se encuentra el cursor.

J Une la línea en la que se encuentra el cursor con la siguiente.

:!<comando> Ejecuta el comando del UNIX dado.

:r !<comando> Lee al *buffer* de edición (en la posición del cursor) lo que escriba por la salida estándar el **comando**.

:d1,d2w archivo Graba en un archivo con el nombre dado, el rango de líneas d1 a d2. Note que este es tan sólo un ejemplo de comando de **ed**. Todos ellos funcionan bajo **vi**.

:d1,d2 !<comando> Envía el rango de líneas d1 a d2 a un "pipe" conectado a la entrada estándar de **comando**. La salida estándar de éste viene a reemplazar el rango de líneas dado. (Sirve para aplicar un filtro a un bloque de líneas).

Q Cambia a modo editor de líneas (**ex**). Es muy similar a trabajar bajo **ed**, sólo que con mayores capacidades. Para retornar a modo visual, dar el comando **vi**.

2.2.12. Selección de preferencias.

vi tiene una serie de parámetros ajustables "al gusto" del usuario. Para ello se usan el comando **:set**. Use, por ejemplo **:set all** para obtener un listado del estado actual de los diversos parámetros ajustables. La mayoría de estos son binarios, es decir, sólo tienen dos estados: habilitado o deshabilitado. A continuación se listan algunas de las opciones más comunes:

autoindent se usa para editar programas. Al colocarlo, cada línea conserva la misma indentación de la línea anterior. Para poder llevar el cursor hacia atrás al principio de una línea, se usa <**ctrl**>**d**. Por naturaleza, se tiene **noautoindent**.

ignorecase si se coloca esta opción, las diversas búsquedas no tienen en cuenta si las palabras están en mayúscula o en minúscula. Por naturaleza se tiene **noignorecase**.

magic hace que **vi** entienda expresiones regulares. Si se deseara que no las entienda (y trate a los caracteres especiales de forma normal) se colocaría **nomagic**. Por naturaleza, se tiene **magic**.

number hace que **vi** numere las líneas en el *buffer* de edición. Por naturaleza se tiene **nonumber**.

shell define cuál es la shell que se llamará al usar el comando **!**. Por naturaleza se tiene **shell=/bin/sh**, que es la shell de Bourne.

wrapscan afecta la forma en que **vi** realiza las búsquedas. Cuando esta característica está puesta las búsquedas proceden desde la posición del cursor hasta el final del buffer y si aún no se halla el patrón buscado, se continúa desde el principio del buffer hasta la posición del cursor. Si se tiene **nowrapscan** la búsqueda termina al final del buffer. El default es **wrapscan**.

Si se desea cambiar en forma permanente alguna de estas características, basta con crear un archivo llamado **.exrc** en el directorio **$HOME** del usuario en el que se incluyan las características deseadas. Por ejemplo, si se deseara tener autoindent y number en forma permanente y además que la shell fuera **csh**, el contenido del archivo .exrc sería:

```
set autoindent
set number
set shell=csh
```

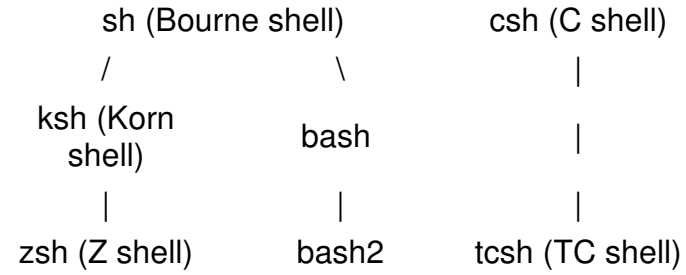
2.3. EJERCICIOS DE REPASO PROPUESTOS

- Sacar una copia del archivo /etc/passwd y de /etc/shadow en el directorio /prueba/curso. Con el editor ed, editarlos y quitar el campo correspondiente a la clave.
- Con la copia hecha al archivo passwd , probar las opciones del editor vi, tratadas.
- Editar los archivos correspondientes al inicio de trabajo de los diferentes usuarios en los shell , para personalizar su ambiente de trabajo. Tener en cuenta cuales son los archivos a modificar dependiendo del shell usado:

2.3.1. Archivos de inicio según el tipo de shell

Definicion de variables en los shell y archivos de inicio de ambiente

La definición de variables varía un poco entre los diferentes shell a utilizar en las máquinas unix.



Hay dos tipos de variables, las variables del shell, que son variables locales a cada shell, y que se pierden cuando se ejecuta un nuevo shell (por ejemplo, si se llama a un script dentro de un shell, o a cualquier otro programa), y las *variables de entorno*, que se pasan a cada uno de los programas que se ejecutan desde el shell.

Las variables locales se definen : set variable=valor y se pueden ver con el comando echo $variable o solo con el comando set. Para eliminar el valor de la variable se usa unset variable.

Las variables de entorno se definen diferente dependiendo del shell:

- En csh o tcsh

  setenv variable valor

  Ejm : setenv TERMDIR $HOME
  setenv PATH $PATH:$HOME/bin
  setenv PATH ${PATH}:DIRECTORY:

  Para remover una variable de entorno :
  unsetenv variable

- En bash , ksh, sh
  variable=valor
  export variable

  o
  export variable=valor

  Ejm: export TERMDIR=$HOME
  export PATH=$PATH:$HOME/bin

Que archivos se procesan al inicio de los diferentes tipos de shell?.
En general los sistemas unix manejan los siguientes archivos:

- csh
  Algunas versiones usan los archivos .cshrc and .login files. Cada versión los ubica en diferentes directorios.

Archivos de inicio:
.cshrc – Siempre al inicio.
.login –Cuando se entra a un shell.
.logout – Al salir de un shell .

- tcsh
  Al inicio
  /etc/csh.cshrc - siempre
  /etc/csh.login - Cuando se entra a un shell.
  .tcshrc -Siempre.
  .cshrc – Si no fue encontrado .tcshrc
  .logout – Al salir de un shell.

- sh

  Al inicio
  /etc/profile –Al inicio de un shell.
  .profile – Personalizado inicio de shell.

- Ksh

  /etc/profile - Al inicio de un shell.
  .profile -. Personalizado inicio de shell.

- bash

  /etc/profile –Al inicio de un shell.
  .bash_profile - Personalizado inicio de shell.
  .profile –Si .bash_profile no esta.
  .bashrc - interactive non-login shells.

3. OTRAS HERRAMIENTAS

Este capítulo contiene información sobre algunos comandos "avanzados", no cubiertos aún en estas notas. Se incluyen comandos de comparación y transformación de archivos, comandos de intercomunicación de usuarios y comandos de backups.

Muchos de estos comandos son muy poderosos y con una gran cantidad de opciones. La descripción de todas y cada una de las diversas opciones de cada uno de los comandos, está por fuera del alcance de estas notas. Se han incluido sólo aquellas que se usan con mayor frecuencia.

3.1. COMANDOS PARA COMPARACION DE ARCHIVOS.

3.1.1. Comando diff.

Este comando muestra por *stdout* las diferencias entre dos archivos de texto. Su sintaxis general es

$**diff [-be] archivo_1 archivo_2**

Supongamos por ejemplo, que se el contenido del archivo **arch1** es

```
La línea 1 esta en los dos archivos
La línea 2 esta en los dos archivos
Esta línea solo esta en el archivo 1
Otra línea que también esta en ambos archivos
Esta línea solo esta en el archivo 1
Otra línea mas en ambos archivos
```

Y el contenido del archivo **arch2** es:

```
La línea 1 esta en los dos archivos
La línea 2 esta en los dos archivos
```

```
Otra línea que también esta en ambos archivos
Esta línea solo esta en el archivo 2
Otra línea mas en ambos archivos
Y otra línea solo en el archivo 2
```

El resultado del comando de comparación entre los dos con diff :

```
$diff arch1 arch2

3d2
< Esta línea solo esta en el archivo 1
5c4
< Esta línea solo esta en el archivo 1
---
> Esta línea solo esta en el archivo 2
6a6,7
> Y otra línea solo en el archivo 2
>
[root@www shell]# diff -e arch1 arch2
6a
Y otra línea solo en el archivo 2

.
5c
Esta línea solo esta en el archivo 2
.
3d
```

Las líneas que inician con un **<** sólo están en el primer archivo. Las que lo hacen con **>**, sólo están en el segundo. Para líneas cambiadas, se mostrarán 2 líneas: la que sólo está en el primer archivo y la que sólo está en el segundo. Las diferencias van precedidas de una línea con el formato **nXm** en donde **n** indica número de línea en el primer archivo, **m**, número de línea en el segundo archivo, y **X** puede ser:

a para indicar líneas que deberían **añadirse** al primer archivo para obtener el segundo,
d para indicar líneas que deberían **borrarse** del primer archivo para obtener el segundo y

c para indicar líneas que deberían **cambiarse** en el primer archivo, para obtener el segundo.

Las opciones pueden ser:

-e La salida muestra los comandos de **ed** que convertirán el primer archivo en el segundo.
-b Indica que no se deben tener en cuenta diferencias debidas a blancos entre las palabras de una línea.

3.1.2. Comando cmp.

Este comando compara byte por byte dos archivos, y tan pronto halla alguna diferencia, la indica. Si se ejecuta con la opción **-l**, mostrará todas las diferencias byte por byte de los dos archivos.

Es útil también para comparar archivos binarios.

Por ejemplo, usando los dos mismos archivos del apartado anterior,

```
$cmp arch1 arch2

arch1 arch2 differ: char 75, line 3
```

$**cmp -l arch1 arch2** da como salida: (sólo se muestra una parte)

```
  73 105 117
  74 163 164
  75 164 162
  84 163 161
  85 157 165
  86 154 145
  87 157  40
  88  40 164
  89 145 141
  90 163 155
  91 164 142
  92 141 151
  93  40 145
  94 145 156
```

95 156 40

(Muestra el número del byte que es diferente y los valores octales del byte en el primero y segundo archivos, respectivamente).

3.1.3. Comando comm.

Este comando determina qué líneas son comunes a dos archivos. El comando produce una salida de tres columnas: la primera contiene todas las líneas que están exclusivamente en el primer archivo, la segunda, las que están exclusivamente en el segundo y la tercera, las que están en ambos. Por ejemplo, y usando los mismos archivos de los ejemplos anteriores:

```
# comm arch1 arch2
                La línea 1 esta en los dos archivos
                La línea 2 esta en los dos archivos
Esta línea solo esta en el archivo 1
                Otra línea que también esta en ambos archivos
Esta línea solo esta en el archivo 1
        Esta línea solo esta en el archivo 2
                Otra línea mas en ambos archivos
        Y otra línea solo en el archivo 2
```

El comando se puede usar con las opciones **-1 -2** ó **-3** para suprimir la salida de la columna 1, 2 o 3 respectivamente.

```
# comm -1 arch1 arch2
        La línea 1 esta en los dos archivos
        La línea 2 esta en los dos archivos
        Otra línea que también esta en ambos archivos
Esta línea solo esta en el archivo 2
        Otra línea mas en ambos archivos
Y otra línea solo en el archivo 2
```

```
# comm -2 arch1 arch2
        La línea 1 esta en los dos archivos
        La línea 2 esta en los dos archivos
Esta línea solo esta en el archivo 1
        Otra línea que también esta en ambos archivos
```

```
Esta línea solo esta en el archivo 1
        Otra línea mas en ambos archivos
```

```
# comm -3 arch1 arch2
Esta línea solo esta en el archivo 1
Esta línea solo esta en el archivo 1
        Esta línea solo esta en el archivo 2
        Y otra línea solo en el archivo 2
```

3.1.4. Comando uniq.

Este comando se usa para eliminar líneas iguales seguidas, en un archivo. Normalmente se usa en un "pipeline" junto con **sort** para eliminar líneas repetidas.

Si, por ejemplo se hace

```
# sort arch1 arch2 >arch3
# cat arch3

Esta línea solo esta en el archivo 1
Esta línea solo esta en el archivo 1
Esta línea solo esta en el archivo 2
La línea 1 esta en los dos archivos
La línea 1 esta en los dos archivos
La línea 2 esta en los dos archivos
La línea 2 esta en los dos archivos
Otra línea mas en ambos archivos
Otra línea mas en ambos archivos
Otra línea que también esta en ambos archivos
Otra línea que también esta en ambos archivos
Y otra línea solo en el archivo 2

#
# uniq arch3

Esta línea solo esta en el archivo 1
Esta línea solo esta en el archivo 2
La línea 1 esta en los dos archivos
```

```
La línea 2 esta en los dos archivos
Otra línea mas en ambos archivos
Otra línea que también esta en ambos archivos
Y otra línea solo en el archivo 2
```

Opciones:
-u Sólo escribe las líneas sin repetir
-d Sólo escribe una copia de las líneas repetidas
-c Como sin opciones, pero cada línea precedida del número de repeticiones.

3.2. COMANDOS DE BUSQUEDA Y DE TRANSFORMACION DE ARCHIVOS.

3.2.1. Comando find.

Este comando busca recursivamente (i.e. explorando subdirectorios) en los directorios dados, los archivos que cumplen con un determinado criterio de búsqueda. En algunas versiones de unix por "default" no produce ninguna salida, sino que el comando indica éxito o fracaso en la búsqueda según el valor del estado del finalización del programa (variable **$?** de la shell).

La sintaxis general del comando es:

$**find lista_de_directorios expresión_booleana**

Para armar la expresión booleana, se pueden usar:

-name archivo	resultado **true** si el archivo existe en el directorio dado.
-perm numero_octal	resultado **true** si existen archivos cuyos permisos (modo) sean exactamente iguales al del número octal dado.
-type tipo	resultado **true** si existen archivos del tipo dado. El tipo puede ser **b** para archivo especial de dispositivo tipo bloque, **c** para dispositivo tipo carácter, **d** para directorio, **f** para archivo regular.
-user nombre	resultado **true** si existen archivos cuyo propietario es el usuario dado por "nombre".

-group nombre resultado **true** si existen archivos cuyo grupo de usuarios es el dado por "nombre".

-newer archivo resultado **true** si existen archivos cuya fecha de modificación es posterior a la del archivo dado.

-print Si la versión por defualt no produce salida, es la única forma en que este comando muestra los resultados. Muestra los path_names de los archivos encontrados.

Los anteriores se pueden combinar con los operadores booleanos:

! negación
-o ó lógico

Combinados sin ningún operador, se sobreentiende "y".

La gran mayoría de las veces, este comando se usa con la opción **-name** y **-print** para encontrar un archivo. Por ejemplo

$**find** / **-name arch1 -print**, buscará en todos los sistemas de archivos el archivo "arch1" y mostrará el pathname del sitio en donde se encuentra.

$**find** /**usr -perm 777 -type d - print**, buscará todos los subdirectorios bajo /**usr** que tienen todos los derechos para todos los usuarios.

```
# find /guillermo  -name "index.html" -print
/guillermo/index.html
```

```
# find /gina -type d -print
/gina
/gina/administracion
/gina/.kde
```

3.2.2. Comando cut.

Este comando se usa para "cortar" columnas (o campos) de cada línea de un archivo dado. Su forma general es:

$**cut [-s] [-clista] [-flista] [-dc] lista_de_archivos**

Las opciones son:

-clista especifica las columnas que deben ser cortadas. Por ejemplo **-c4-15,20-40**, especifica que se deben cortar las columnas 4 a la 15 y 20 a la 40.

-flista especifica que lo que se deben cortar no son columnas sino campos (los dados). Por ejemplo **-f3-5** indica que se deben cortar los campos 3, 4 y 5. El delimitador, si no se dice otra cosa, es el carácter Tab (pero se puede cambiar con la opción **-c**). Las líneas que no contengan delimitadores son pasadas intactas, a no se que se use la opción **-s**.

-dc especifica que debe usarse como delimitador, el carácter **c**.

-s especifica que las líneas que no tengan delimitadores, deben suprimirse del archivo de salida.

Ejemplos:

$**cut -f1,5 -d: -s /etc/passwd**, mostrará los campos 1 y 5 del archivo /etc/passwd, usando como delimitador el carácter **:**. Estos campos corresponden al nombre del usuario y a su descripción.

```
# cut -f 1 -d: /etc/passwd
root
bin
daemon
adm
hector
webmaster
jhoana
gina
guillermo
sayi
carolina
```

```
 # cut -f 1,7 -d: /etc/passwd
root:/bin/bash
bin:/sbin/nologin
daemon:/sbin/nologin
```

```
adm:/sbin/nologin
hector:/bin/bash
webmaster:/bin/bash
jhoana:/bin/bash
gina:/bin/bash
guillermo:/bin/bash
sayi:/bin/bash
carolina:/bin/bash
```

```
# cut -c 1,7  /etc/passwd
r:
b1
d:
a3
h:
wt
j:
g:
gr
s:
cn
```

```
# cut -c 1-7  /etc/passwd
root:x:
bin:x:1
daemon:
adm:x:3
hector:
webmast
jhoana:
gina:x:
guiller
sayi:x:
carolin
```

```
# ls -l / |cut -c2-10,55-
otal 240
```

```
rwxr-xr-x bin
rwxr-xr-x boot
rwxr-xr-x capturas
rwxr-xr-x carolina
rwxr-xr-x descargas
rwxr-xr-x dev
rwxr-xr-x etc
rw------- filtros
rwxr-xr-x gina
rwxr-xr-x guillermo
rwxr-xr-x home
rwxr-xr-x initrd
rwxr-xr-x jhoana
rwxr-xr-x lib
rwxr-xr-x lost+found
rwxr-xr-x misc
rwxr-xr-x mnt
rwxr-xr-x opt
r-xr-xr-x proc
rw-r--r-- prueba
rwxr-xr-x pruebadir
rw-r--r-- pruebaftp
rwxr-xr-x pruebap
rwxr-x--- root
rwxr-xr-x sayi
rwxr-xr-x sbin
rw-r--r-- sendmail.txt
rwxr-xr-x tftpboot
rwxrwxrwt tmp
rwxr-xr-x usr
rwxr-xr-x var
rwxr-xr-x varios
rwxrwxrwx web -> /etc/httpd/conf/httpd.conf
rwxr-xr-x windows
```

$**ls -l** / |**cut -c1,10-12,55-80**, produciría un listado del directorio raíz en el que se mostraría el tipo del archivo (columna 1), espacio en blanco (columnas 11 a la 13) y el nombre del archivo (columnas 55 a la 80). Un ejemplo de salida es:

```
#ls -l / |cut -c1,10-12,55-80
```

```
t
-  LOGO
d  bin
d  dev
d  etc
d  lib
d  lost+found
-  mbox
-  mkfs.data
d  mnt
-  nohup.out
d  shlib
d  tmp
-  unix
d  usr
d  usr2
```

3.2.3. Comando sort.

Este comando ordena las líneas de los archivos dados y produce su salida por *stdout*. El ordenamiento se hace comparando las líneas de acuerdo a una o más claves de ordenamiento. El "default" es una sola clave de ordenamiento, que es toda la línea. Cada campo se considera separado por un carácter de tabulado o un blanco.

Este comando tiene una gran cantidad de opciones y es algo complicado de entender. Sobre todo la explicación de las opciones. Se verán aquí las básicas:

$**sort [-nrbt] [+m.n] [-o.p] lista_de_archivos**

En nuevas versiones existe una sintaxis mas sencilla:

$ sort [-nrbt] [-k #] lista_de_archivos

Las opciones son:

-n El ordenamiento debe hacerse numéricamente, no lexicográficamente.
-r El ordenamiento debe ser reverso, es decir descendente.
-b Ignorar blancos al principio de los campos

-t<car> Los separadores de campos son cambiados por el carácter <car>
-k # **Indica el campo que se tomara como llave**

Las claves de ordenamiento se dan con el formato **+m.n -o.p**, que significan "ordenar desde el primer carácter + **n** del campo primero + **m** hasta el **p**-ésimo carácter después del último carácter del campo **o**". Si cualquiera, **n** o **p** es cero, no es necesario colocarla. Si no se coloca **-o.p** se sobreentiende que es hasta el final de la línea. El **m** se puede combinar con cualquiera de las opciones anteriores, para indicar que tal opción sólo aplica a esta clave de ordenamiento. (Hay que admitir que la forma de dar las claves de ordenamiento es un poco "extraña", por decir lo menos).

Ejemplos:

$**sort mi_archivo**, ordenará ascendentemente las líneas del archivo "mi_archivo", tomando como clave de ordenamiento toda la línea.

$**sort +2 -5 datos_1 datos_2>datos_t**, ordenará el contenido de los dos archivos "datos_1" y "datos_2" dejando el resultado en "datos_t". El criterio de ordenamiento será: desde el primer carácter (ya que **n** no está dado) del **tercer** (primero más 2) campo hasta el último carácter (ya que **p** no está dado) del quinto campo. Si se deseara este mismo criterio, pero en orden descendente, el comando sería **sort +2r -5 ...**.

$**sort +1 -2 +4r -5 arch1**, ordenaría primero en orden ascendente de acuerdo al contenido completo del campo 2 y las líneas que tengan igual el campo 2, quedarán ordenadas descendentemente según el contenido completo del campo 5.

Suponga ahora que se desea producir un listado del archivo /etc/passwd, ordenado por el número de identificación del usuario (es el tercer campo de cada línea, usando **:** como delimitador). El comando sería:

$**sort -t: +2n -3 /etc/passwd**. Note que se colocó la bandera **n** para indicar que se desea orden numérico y no lexicográfico

Finalmente, suponga que es necesario ordenar el archivo "file1" en forma ascendente usando como criterio únicamente el segundo carácter del tercer campo. El comando sería:

$**sort +2.1 -3.2 file1**

sort -t: -k 1 -r /etc/passwd

```
xfs:x:43:43:X Font Server:/etc/X11/fs:/bin/false
webmaster:x:501:501::/home/webmaster:/bin/bash
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
squid:x:23:23::/var/spool/squid:/dev/null
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
sayi:x:505:505::/sayi:/bin/bash
rpm:x:37:37::/var/lib/rpm:/bin/bash
rpc:x:32:32:Portmapper RPC user:/:/bin/false
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
root:x:0:0:root:/root:/bin/bash
radvd:x:75:75:radvd user:/:/bin/false
pvm:x:24:24::/usr/share/pvm3:/bin/bash
postgres:x:26:26:PostgreSQL Server:/var/lib/pgsql:/bin/bash
pcap:x:77:77::/var/arpwatch:/bin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
nscd:x:28:28:NSCD Daemon:/:/bin/false
nobody:x:99:99:Nobody:/:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
news:x:9:13:news:/var/spool/news:
named:x:25:25:Named:/var/named:/bin/false
mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
mailnull:x:47:47::/var/spool/mqueue:/dev/null
mailman:x:41:41:GNU Mailing List Manager:/var/mailman:/bin/false
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
ldap:x:55:55:LDAP User:/var/lib/ldap:/bin/false
junkbust:x:73:73::/etc/junkbuster:/bin/bash
jhoana:x:502:502::/jhoana:/bin/bash
ident:x:98:98:pident user:/:/sbin/nologin
hector:x:500:500::/home/hector:/bin/bash
halt:x:7:0:halt:/sbin:/sbin/halt
guillermo:x:504:504::/guillermo:/bin/bash
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
gina:x:503:503::/gina:/bin/bash
gdm:x:42:42::/var/gdm:/sbin/nologin
```

```
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
carolina:x:506:506::/carolina:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
apache:x:48:48:Apache:/var/www:/bin/false
amanda:x:33:6:Amanda user:/var/lib/amanda:/bin/bash
adm:x:3:4:adm:/var/adm:/sbin/nologin
```

```
# sort -t: -k 1  /etc/passwd
adm:x:3:4:adm:/var/adm:/sbin/nologin
amanda:x:33:6:Amanda user:/var/lib/amanda:/bin/bash
apache:x:48:48:Apache:/var/www:/bin/false
bin:x:1:1:bin:/bin:/sbin/nologin
carolina:x:506:506::/carolina:/bin/bash
daemon:x:2:2:daemon:/sbin:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gdm:x:42:42::/var/gdm:/sbin/nologin
gina:x:503:503::/gina:/bin/bash
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
guillermo:x:504:504::/guillermo:/bin/bash
halt:x:7:0:halt:/sbin:/sbin/halt
hector:x:500:500::/home/hector:/bin/bash
ident:x:98:98:pident user:/:/sbin/nologin
jhoana:x:502:502::/jhoana:/bin/bash
junkbust:x:73:73::/etc/junkbuster:/bin/bash
ldap:x:55:55:LDAP User:/var/lib/ldap:/bin/false
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
mailman:x:41:41:GNU Mailing List Manager:/var/mailman:/bin/false
mailnull:x:47:47::/var/spool/mqueue:/dev/null
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash
named:x:25:25:Named:/var/named:/bin/false
news:x:9:13:news:/var/spool/news:
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
nscd:x:28:28:NSCD Daemon:/:/bin/false
ntp:x:38:38::/etc/ntp:/sbin/nologin
```

```
operator:x:11:0:operator:/root:/sbin/nologin
pcap:x:77:77::/var/arpwatch:/bin/nologin
postgres:x:26:26:PostgreSQL Server:/var/lib/pgsql:/bin/bash
pvm:x:24:24::/usr/share/pvm3:/bin/bash
radvd:x:75:75:radvd user:/:/bin/false
root:x:0:0:root:/root:/bin/bash
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
rpc:x:32:32:Portmapper RPC user:/:/bin/false
rpm:x:37:37::/var/lib/rpm:/bin/bash
sayi:x:505:505::/sayi:/bin/bash
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
squid:x:23:23::/var/spool/squid:/dev/null
sync:x:5:0:sync:/sbin:/bin/sync
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
webmaster:x:501:501::/home/webmaster:/bin/bash
xfs:x:43:43:X Font Server:/etc/X11/fs:/bin/false
```

```
# sort -t: -k 3 -n /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/var/spool/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
squid:x:23:23::/var/spool/squid:/dev/null
pvm:x:24:24::/usr/share/pvm3:/bin/bash
named:x:25:25:Named:/var/named:/bin/false
postgres:x:26:26:PostgreSQL Server:/var/lib/pgsql:/bin/bash
mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash
nscd:x:28:28:NSCD Daemon:/:/bin/false
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
```

```
rpc:x:32:32:Portmapper RPC user:/:/bin/false
amanda:x:33:6:Amanda user:/var/lib/amanda:/bin/bash
rpm:x:37:37::/var/lib/rpm:/bin/bash
ntp:x:38:38::/etc/ntp:/sbin/nologin
mailman:x:41:41:GNU Mailing List Manager:/var/mailman:/bin/false
gdm:x:42:42::/var/gdm:/sbin/nologin
xfs:x:43:43:X Font Server:/etc/X11/fs:/bin/false
mailnull:x:47:47::/var/spool/mqueue:/dev/null
apache:x:48:48:Apache:/var/www:/bin/false
ldap:x:55:55:LDAP User:/var/lib/ldap:/bin/false
junkbust:x:73:73::/etc/junkbuster:/bin/bash
radvd:x:75:75:radvd user:/:/bin/false
pcap:x:77:77::/var/arpwatch:/bin/nologin
ident:x:98:98:pident user:/:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
hector:x:500:500::/home/hector:/bin/bash
webmaster:x:501:501::/home/webmaster:/bin/bash
jhoana:x:502:502::/jhoana:/bin/bash
gina:x:503:503::/gina:/bin/bash
guillermo:x:504:504::/guillermo:/bin/bash
sayi:x:505:505::/sayi:/bin/bash
carolina:x:506:506::/carolina:/bin/bash
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
```

3.2.4. Comando grep.

Este comando tiene el propósito de imprimir (por stdout) las líneas que cumplen con una expresión regular dada dentro de una serie de archivos (grep es una sigla de **global r**egular **e**xpresion **p**rinter).

El formato general es

$**grep [-v] expresion_regular lista_de_archivos**

La opción **-v** significa que invierta su funcionamiento normal, es decir que liste las líneas que **no cumplen** con la expresión regular dada.

Ejemplos:

$**grep stdio /usr/hcg/fuentes/*.c**, listará todas las líneas que contengan la cadena "stdio" de todos los archivos cuyo nombre termine en ".c" en el directorio /usr/hcg/fuentes. Note que "stdio" no fue necesario encerrarla entre comillas entre el comando, ya que no tiene ningún carácter que pueda ser expandido por la *shell*.

$**grep "^#" *.sh**, listará todas las líneas que empiezan (**^**) con el carácter "#", de todos los archivos cuyo nombre termina en ".sh".

$**grep " .... " archivo1**, listará todas las líneas del archivo1, que contienen palabras de 4 letras.

$**grep "^QQ[0-9]{5}[A-Z]" /usr/inventario/partes**, listará todas las líneas del archivo /usr/inventario/partes, que empiezan con "QQ" seguido de un número de 5 dígitos ([0-9] repetido cinco veces, {5}) y que a continuación traen una letra mayúscula.

Existe también el comando **fgrep** (fast grep) que funciona en forma muy similar al grep, pero sólo admite "strings" en los que no reconoce los caracteres especiales de expresiones regulares.

Por ejemplo deseo saber en que archivos del directorio /etc/xinetd.d se hace referencia a la palabra telnet

```
# grep telnet /etc/xinetd.d/*
/etc/xinetd.d/krb5-telnet:# description: The kerberized telnet server accepts normal
telnet sessions, \
/etc/xinetd.d/krb5-telnet:service telnet
/etc/xinetd.d/krb5-telnet:      server          = /usr/kerberos/sbin/telnetd
/etc/xinetd.d/telnet:# description: The telnet server serves telnet sessions; it uses \
/etc/xinetd.d/telnet:service telnet
/etc/xinetd.d/telnet:   server          = /usr/sbin/in.telnetd
```

3.3. UTILIDADES PARA COMUNICACION ENTRE USUARIOS.

3.3.1. Utilidad mail.

Los sistema UNIX proveen varias utilidades para comunicación entre usuarios, siendo la más común **mail**. De ella existen versiones más o menos diferentes y más o menos poderosas. En estas notas se describirá el **mail** básico.

Para enviar correo se usa el comando **mail** así:

$**mail [-t] lista_de_usuarios**

El programa leerá el mensaje deseado por *stdin* y lo enviará a cada uno de los usuarios listados en la invocación el comando (Cada usuario en el sistema tiene un "buzón de correo" en el que son depositados los mensajes que le han sido enviados). Cada mensaje es precedido de una "marca de correo" que indica quién lo envió, en qué fecha y el destinatario.

La opción **-t** indica que el mensaje sea además precedido de una lista de todos los usuarios a los que se envió el mensaje.

Por ejemplo

$**mail hgt <memo**

enviará el contenido del archivo "memo" al usuario hgt.

Para leer el correo recibido se usa **mail** sin opciones. Si hay correo en el buzón del usuario, el programa mostrará cada mensaje, empezando por el más reciente. Después de mostrar cada mensaje, el programa coloca un **?** y espera un comando que le diga qué hacer con el mensaje acabado de leer. Si se contesta con <enter>, mostrará el siguiente mensaje. Otras respuestas pueden ser:

+	Ver siguiente mensaje
d	Borrar el mensaje corriente y ver el siguiente
p	Mostrar nuevamente el mismo mensaje
-	Mostrar el mensaje anterior
s [archivo]	salvar el mensaje en el archivo dado (y se borra del buzón)
w [archivo]	como el anterior, pero sin el encabezamiento (y se borra del buzón)
m [usuarios]	enviar el mensaje a los usuarios dados (y se borra del buzón)

q salir de **mail**.
x salir de **mail**. Todos los mensajes (aún los "borrados") se colocan nuevamente en el buzón).
! comando Ejecuta el "comando" dado
? muestra una lista con estas opciones.

3.3.2. Comando write.

Este comando permite enviar mensaje a usuarios que estén en línea. Su formato general es

$**write usuario [línea]**

Una vez invocado, el comando lee por *stdin* el mensaje a ser enviado. La opción de **línea** se usa en el caso de que el mismo nombre de usuario haya entrado en más de una terminal. Así puede indicarse a cuál de las varias "instancias" del usuario debe enviarse el mensaje.

El usuario al que se escribe el mensaje verá en su pantalla un aviso del usuario y la fecha y hora en que se empezó a enviar el mensaje y luego el contenido el mensaje.

Si un usuario desea no recibir mensajes, puede usar el comando **mesg -n** (**mesg -y**, si quiere aceptar mensajes de nuevo).

3.3.3. Comando wall.

Este es un comando que sólo puede ser ejecutado por el "superusuario" y permite enviar un mensaje a todos los usuarios que se encuentren trabajando en el sistema. Como los anteriores, el mensaje a ser enviado es leído por stdin.

Por ejemplo:

```
$wall
El sistema se bajará en 10 minutos. Por favor, salve su trabajo y salga o
arriésguese a perder sus archivos!
<ctrl>-d
```

(Note que <ctrl>-d es el indicador de "fin de archivo"

3.4. RESPALDO EN MEDIOS MAGNETICOS.

Estos temas se trataran con más detalle en la parte de administraión de los respectivos sistemas operativos.

3.4.1. Comando tar.

El UNIX tiene un sistema automatizado de backups incrementales muy completo. Normalmente hay un administrador del sistema que se encarga de mantener copias de respaldo apropiadas.

Sin embargo un usuario puede requerir de una copia de algunos de sus archivos, bien sea como respaldo o para transportar información a otra máquina. Hay varias formas de lograr este propósito, siendo tal vez la más sencilla el comando **tar** (**t**ape **ar**chive). La copia puede hacerse a cinta o a diskettes. La forma general de este comando es:

$**tar opciones lista_de_directorios_o_de_archivos**

Las opciones son:

c Copia al dispositivo destino los archivos dados (sobreescribe cualquier contenido anterior)
x Extrae del dispositivo (cinta o disco) los archivos pedidos
t Muestra la tabla de contenido de una cinta o un disco obtenido con **tar**
r Reemplaza archivos en una cinta o disco previamente grabado con **tar**. Si el archivo dado ya existía lo reemplaza y si no existía la añade. Esta opción suele ser muy lenta.

Toda invocación de **tar** debe tener una (y sólo una) de las opciones anteriores. Las demás opciones son:

v Normalmente **tar** trabaja "silenciosamente". Si se añade esta opción el programa irá informando los archivos que ha copiado o restaurado.
w Con esta opción, **tar** pedirá confirmación antes de escribir cualquier archivo.

f Indica que el siguiente parámetro es el nombre del dispositivo (en /dev) al que se debe realizar la copia. El algunos UNIX, puede usarse en cambio un número que hace referencia a una línea en el archivo /etc/default/tar, para indicar cuál es el dispositivo para la copia.

Ejemplos:

$**tar cv .**, copiará todos los archivos del directorio corriente al dispositivo default.

$**tar cvf /dev/rct0 /usr/pedro /usr/juan**, producirá una copia de todos los archivos de /usr/pedro y de /usr/juan al dispositivo /dev/rct0.

$**tar xv**, extraerá todos los archivos en el dispositivo defaut. (Si se no especifican nombres de archivos a restaurar, **tar** sobreentiende todos). Si la copia se hizo dando pathnames absolutos, el restaure se hará a los mismos pathnames. Si la copia se hizo usando pathnames relativos, el restaure se hará a partir del directorio corriente.

```
# tar cvf /dev/rct0 /gina
tar: Removing leading `/' from member names
gina/
gina/administracion/
gina/administracion/backgrnd.gif
gina/administracion/administracion.html
gina/administracion/ciclo1.php
gina/administracion/ciclo.html
gina/administracion/formulario.html
gina/administracion/index.html
gina/administracion/vaquita.gif
gina/administracion/inserciclo1.php
gina/administracion/inserciclo.html
gina/administracion/inserformulari1.php
gina/administracion/modiciclo0.php
gina/administracion/modiciclo1.php
gina/administracion/clave
gina/administracion/.htaccess
gina/aplicacion.html
gina/ciclo1.php
gina/ciclo.html
```

```
gina/index.html
```

Para observar elcontenido de la copia en medio magnetico sin leerla a disco (sacar un indice del medio):

```
# tar tvf /dev/rct0
drwxr-xr-x gina/root         0 2003-03-23 15:21:46 gina/
drwxr-xr-x gina/root         0 2003-03-23 15:17:19 gina/administracion/
-rw-r--r-- gina/gina    103582 2003-03-23 12:49:02
gina/administracion/backgrnd.gif
-rw-r--r-- gina/gina       554 2003-03-23 15:17:21
gina/administracion/administracion.html
-rwxr-xr-x gina/root      2043 2003-03-05 09:51:30 gina/administracion/ciclo1.php
-rwxr-xr-x gina/root      1197 2003-03-05 09:51:30 gina/administracion/ciclo.html
-rwxr-xr-x gina/root      4862 2003-03-23 12:41:46
gina/administracion/formulario.html
-rwxr-xr-x gina/root       575 2003-03-23 14:32:45 gina/administracion/index.html
-rw-r--r-- gina/gina      3808 2003-03-23 12:50:22 gina/administracion/vaquita.gif
-rwxr-xr-x gina/root      1076 2003-03-23 13:20:11
gina/administracion/inserciclo1.php
-rwxr-xr-x gina/root      1423 2003-03-23 13:16:27
gina/administracion/inserciclo.html
-rwxr-xr-x gina/root      2169 2003-03-23 13:05:15
gina/administracion/inserformulari1.php
-rwxr-xr-x gina/root      1985 2003-03-23 13:32:02
gina/administracion/modiciclo0.php
-rwxr-xr-x gina/root       862 2003-03-23 13:35:52
gina/administracion/modiciclo1.php
-rwxr-xr-x gina/root        21 2003-03-05 09:57:58 gina/administracion/clave
-rwxr-xr-x gina/root       108 2003-03-05 09:57:12 gina/administracion/.htaccess
-rwxr-xr-x gina/root      1732 2003-03-23 11:12:20 gina/aplicacion.html
-rwxr-xr-x gina/root      2111 2003-03-23 10:07:33 gina/ciclo1.php
-rwxr-xr-x gina/root       986 2003-03-23 09:55:34 gina/ciclo.html
-rwxr-xr-x gina/root      1117 2003-03-23 09:34:54 gina/index.html
```

3.5. EJERCICIOS DE REPASO PROPUESTOS.

- Escribir un comando que presente por pantalla, los procesos del usuario root, ordenados por PID de forma descendente y que solo presente las columnas del nombre del usuario, PID, PPID, y comando ejecutado.
- Presentar en pantalla , los 20 archivos o directorios que mas espacio ocupan en / , de mayor a menor. Si hay más sistemas de archivos , tales como /home, /tmp, /proc, no se desea que presente información de estos. Solo de la partición /.
- Crear varios archivos dentro del directorio /compartido, con cualquier contenido. Darle permisos 777 a estos.
- Presentar un listado de los archivos que consideramos inseguros en el sistema (pues tienen permisos 777).
- Repetir el ejercicio anterior, pero en el mismo comando indicarle al sistema que borre esos archivos.
- Suponiendo que el usuario pruebas , se retirará del sistema, se desea hacer una limpieza de sus archivos. Se va a examinar en toda la máquina que archivos son de el y proceder a borrarlos dentro del mismo comando.
- Sacar un reporte ordenado por nombre de usuario, de los usuarios que no tienen shell asignado. Solo se desea ver el nombre del usuario, el UID, el directorio HOME .
- Sacar un reporte ordenado por nombre de usuario, de los usuarios que trabajan con bash . Solo se desea ver el nombre del usuario, el UID, el directorio HOME y el shell.
- Sacar una copia del directorio /compartido en un dispositivo ficticio llamado /dev/cinta.
- Restaurar la copia realizada en el dispositivo /dev/cinta en /todos.

4. PROGRAMACION SHELL

Ya se ha estudiado la shell como un interpretador de comandos. En este curso se verán todas las facilidades que tiene la shell para realizar programas. Gracias a ellas es posible escribir programas que se "construyen" a partir de otros programas del sistema operacional.

4.1. BASES

Debemos recordar algunos conceptos necesarios para la ejecución de comandos y uso de Metacaracteres.

4.1.1. METACARACTERES

Los metacaracteres son caracteres especiales que representan otros caracteres. Son algunas veces llamados "wildcards" o comodines. Son usados para generar nombres que concuerden con nombres de archivos o parte de nombres de archivos simplificando la tarea de especificar archivos o grupos de archivos como argumentos de los comandos.

En esta forma se simplifican tareas tales como borrar de un directorio todos los archivos que terminen en .c

Es importante entender que estos metacaracteres son usados por la shell para generar nombres de archivos basada en el significado del carácter, comparado contra el contenido del directorio especificado.

Los metacaracteres de la shell son:

*****: Concuerda con cualquier string de caracteres incluyendo el string nulo. Usted puede usar el * para especificar un nombre de archivo total o parcial.

?: Concuerda con un solo carácter de un nombre de archivo, pero puede ser usado más de una vez.

[]: Cuando desea que el shell concuerde con uno cualquiera de varios posibles caracteres. Estos caracteres pueden aparecer en cualquier posición en el nombre del archivo.

También es posible especificar rangos de letras o dígitos. separándolos con un guión. Por ejemplo, si se especifica capitulo[1-3] concordará con nombres como capitulo1, capitulo2 y capitulo3.

Ejemplos:

$ ls -x h*****

hector hector1 hector2 hector3 hector4 hector5

$ ls -x hector**?**

hector1 hector2 hector3 hector4 hector5

$ ls -l hector**[2-4]**

hector2 hector3 hector4

Los diversos metacaracteres se pueden mezclar en sólo comando. El siguiente comando, por ejemplo, copiaría todos los archivos cuyo nombre empiece por una letra mayúscula y vaya seguido de almenos un carácter:

$cp **[A-Z]?*** /usr/people/hector/copias

Cómo interpreta la shell los metacaracteres?

Para cada archivo que cumpla el criterio de selección (dado por los metacaracteres), la shell crea un argumento para el comando. Luego la shell arma el comando tal como si el usuario no hubiera usado metacaracteres, sino hubiera digitado los nombres completos. En esta forma, el comando que recibe los argumentos no "sabe" si el usuario usó o no metacaracteres y por lo tanto no es necesario que cada

programa (comando) tenga el código necesario para manejar estos caracteres especiales. oda esta función está centralizada en la shell.

4.1.2. CORRIENDO PROGRAMAS EN HORA DIFERENTE

Los comandos **batch** y **at** permiten especificar un comando o secuencias de comandos para ser ejecutados más tarde.

4.1.2.1. Batch.

En este comando, es el sistema quién determina a que hora se ejecutará el comando. El sistema le asigna un número a ese trabajo . El formato del comando es :

batch
comando1
comando2
....
<crtl-D>

Si se tiene un solo comando su formato es:

batch comando
<crtl-D>

Ejemplo:

batch
pwd >salida
date >>salida
ls -l >>salida
^D

job 729961819.b at Wed Feb 17 10:10:19 1993

4.1.2.2 AT

Con este comando, es el usuario quién determina a que hora se ejecutará su trabajo. El sistema le asigna un número a ese trabajo. El formato es :

at tiempo
  comando1
  comando2
  ....
<ctrl-D>

donde tiempo puede estar en el formato :

8:15am Feb 20 ó
8:15pm Feb 20
8:12pm

Para observar y cancelar un trabajo en espera de ser procesado se tiene:

at -l	Lista los trabajos en cola
at -r jobnum	Cancela el trabajo identificado con el numero "jobnum".

Ejemplo:

```
# at 10:15am Feb 17
  date >salidaat
  ps -ef >>salidaat
 ^D

job 729962100.a at Wed Feb 17 10:15:00 1993


# date
Wed Feb 17 10:12:28 EST 1993

# at -l
```

user = root 729962100.a Wed Feb 17 10:15:00 1993

Se puede enviar una secuencia de comandos, indicándole que lo lea de un archivo. En este caso batch_at

Ejemplo:

at 10:20am Feb 18 <batch_at

job 730048800.a at Thu Feb 18 10:20:00 1993

at -l

user = root 729962100.a Wed Feb 17 10:15:00 1993
user = root 730048800.a Thu Feb 18 10:20:00 1993

Para cancelar un trabajo :

at -r 730048800.a

4.1.3. COMANDO NOHUP

Todos los procesos de un usuario son terminados cuando este sale del sistema (hace logout). Si se desea que un proceso en background (recordar que para correr un comando o programa en background se debe colocar al final de la línea el carácter &) continué corriendo después de que el usuario salga del sistema, se debe usar el comando **nohup**. El formato de la línea es:

nohup comando **&**

Se puede terminar el comando con el **Kill**

4.2. PROGRAMACION DE LA SHELL

La Shell es un interpretador de comandos, como se ha estudiado hasta ahora. Pero se puede usar para crear programas o nuevos comandos; los cuales son llamados "Procedimientos Shell" o "Shell Scripts". Estos son archivos de comandos que se pueden ejecutar.

Supongamos tenemos el archivo **lista** con los comandos:

```
$ cat lista
  ls -l
  date
  pwd
```

Para ejecutarlos se usa el comandos **sh**, así :

```
$sh lista

total 21

-rw-r--r--   1 hector   other    19 Feb 17 10:33 lista
-rw-r--r--   1 hector   other   694 Feb 17 10:32 listado
-rwxrwxrwx   1 hector   other   413 Feb 17 10:07 menu
-rw-r--r--   1 hector   other     0 Feb 17 10:33 pgma.doc
-rw-r--r--   1 hector   other   441 Feb 17 10:10 salida
-rwxrwxrwx   1 hector   other   52  Feb 17 10:23 shell.sh

Wed Feb 17 10:33:54 EST 1993

/usr/people/hector/cursoshell
```

Este comando también invoca una subshell que lee los comandos del archivo "lista".

Si se desea ejecutar como un comando cualquiera se debe tener en cuenta :

- Cambiar los derechos del archivo para que se pueda ejecutar. Dar derechos **x** al dueño y a otros si se desea. En este caso en **lista** el usuario solo tiene derechos rw y si se trata de ejecutar ocurrirá:

$ls -l lista

-**rw**-r--r-- 1 hector other 19 Feb 17 10:33 lista

$ **./**lista
./lista: cannot execute

Se debe dar derechos x :

$ **chmod u+x** lista

$ ls -l lista

-**rwx**r--r-- 1 hector other 19 Feb 17 10:33 lista

$ **./**lista

- En el caso anterior lo estamos ejecutando desde el directorio en uso (**./**). Si se desea ejecutar desde cualquier directorio se debe incluir en el PATH, pues si se invoca simplemente ocurrirá:

$ lista

lista: not found

$ echo $PATH

/usr/people/hector/bin:/usr/net:/usr/bin:/bin:/usr/ucb:/usr/etc:/etc:/usr/lbin:/usr/new.:/usr/people/hector/bin

$ pwd

/usr/people/hector/cursoshell

El directorio actual no esta en el PATH y se debe incluir allí :

$ PATH=$PATH:/**usr**/**people**/**hector**/**cursoshell**

$ echo $PATH

/usr/people/hector/bin:/usr/net:/usr/bin:/bin:/usr/ucb:/usr/etc:/etc:/usr/lbin:/usr/new.:/usr/people/hector/bin:/**usr**/**people**/**hector**/**cursoshell**

$ lista

Ahora si se puede ejecutar desde cualquier directorio. Lo más conveniente es crear un subdirectorio con un nombre determinado para guardar todos los Script creados por el usuario y adicionar este al PATH en el archivo **.profile**.

4.2.1. ARGUMENTOS DE LA LINEA DE COMANDOS

Las variables que son pasadas a una shell en el momento de la invocación se denominan "argumentos de la línea de comandos" ó "parámetros posicionales". Dentro del procedimiento se hace referencia a ellas mediante un número que indica su posición. Así $1, hará referencia al primer parámetro, $2 al segundo, y así sucesivamente hasta $9. Ejemplo:

$ cat pp

```
#Prueba de parametros posicionales
echo  EL PRIMER PARAMETRO POSICIONAL ES : $1
echo  EL SEGUNDO PARAMETRO POSICIONAL ES : $2
echo  EL TERCER PARAMETRO POSICIONAL ES : $3
```

$ **pp** uno dos tres

```
EL PRIMER PARAMETRO POSICIONAL ES : uno
EL SEGUNDO PARAMETRO POSICIONAL ES : dos
EL TERCER PARAMETRO POSICIONAL ES : tres
```

Además de las variables predefinidas por el sistema, de las definidas por el usuario y de las correspondientes a los parámetros posicionales, la shell posee las siguientes:

$? Esta variable contiene el "valor de salida" del último comando ejecutado. Todo comando en UNIX (aún los "shell scripts" escritos por el usuario, terminan con un determinado "valor de salida". Se ha convenido que ese valor sea cero si el programa termina exitosamente y diferente de cero si no.

$$ Esta variable contiene el número de identificación de proceso de la shell que está siendo ejecutada (el PID).

$! Contiene el PID del último proceso en background invocado por la shell.

$# Contiene el número de argumentos (parámetros posicionales) con que se invocó la shell.

$* Es una sola palabra que contiene todos los argumentos con que se invocó la shell, es decir, equivale a "$1 $2 $3 $4 ...". Se usa normalmente para pasar toda la lista de argumentos a otro programa.

$@ Contiene la lista de todos los argumentos, pero mantenidos como palabras independientes, equivalente a "$1" "$2" "$3" ...

Ejemplos:

$ cat get.num

prueba de numero de argumentos
echo EL NUMERO DE ARGUMENTOS ES: **$#**

$ **get.num** uno dos tres cuatro cinco

EL NUMERO DE ARGUMENTOS ES: 5

$ cat show.param

#prueba de Mostrar todos los parametros
echo LOS PARAMETROS DE ESTE COMANDO SON : **$***

$ **show.param** uno dos tres cuatro cinco seis

LOS PARAMETROS DE ESTE COMANDO SON : uno dos tres cuatro cinco seis

$ **show.param** 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

LOS PARAMETROS DE ESTE COMANDO SON : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

$ ls -x s*

salida salidaat shell.sh show.param

$ **show.param** s*

LOS PARAMETROS DE ESTE COMANDO SON : salida salidaat shell.sh show.param

4.2.2. CONDICIONALES SIMPLES

En todo programa se requiere la capacidad de tomar decisiones. La shell provee varias formas de toma de decisiones, siendo los condicionales simples una de ellas.

Hay dos condicionales simples:

Operador **&&**. Su uso se muestra mejor en el siguiente ejemplo:

$ls -d /usr/people/hector>dev/null **&&** echo Existe /usr/people/hector

El condicional **&&** permite ejecutar el comando a continuación del operador (el **echo** en el ejemplo) si el estado de salida (**$?**) del comando anterior es 0 (que en la shell es "**true**"). El comando **ls** termina con un estado de salida **0** si logró listar algo, de modo que el ejemplo anterior escribirá "Existe

/usr/people/hector" en la pantalla, si existe el directorio /usr/people/hector.

Operador ||. Permite ejecutar el comando (o serie de comandos) a continuación del operador, sólo si el estado de salida del comando anterior es **diferente de 0** (que en la shell es "**false**"). Por ejemplo:

$ls -d /usr/hector || Echo No existe /usr/hector

4.2.3. VARIABLES

Otro tipo de variables que se pueden usar dentro de un programa shell, son las variables a las que el usuario les asigna algún valor. Este tipo de asignación se puede realizar de diferentes maneras:

- Asignación directa.
- Usando el comando **read**.
- Redireccionando la salida de un comando a la variable.
- Asignándole un parámetro posicional.

4.2.3.1.Asignación directa.

Almacenando en la variable un valor predefinido por el usuario. Ejemplo:

TERM=pt200

4.2.3.2. Usando el comando read.

Permite preguntar al usuario el valor deseado para la variable, o las variables . El formato general es:

read variable1 variable2 ...

Ejemplo:

$ cat mknum

#Prueba de asignar valor a variable con Read
echo Digite nombre:
read name
echo Digite codigo:
read cod
echo Digite Telefono y Ciudad
read tel city
echo Los valores leidos son : $name $cod $tel $city

$ mknum

Digite nombre:
Hector
Digite codigo:
7777
Digite Telefono y Ciudad
6101050 Bogota
Los valores leidos son : Hector 7777 6101050 Bogota

4.2.3.3. Redireccionando salida de un comando.

Se puede sustituir la salida de un comando por el valor de la variable usando :

variable**=**`comando`

Ejemplo:

$ cat t

#Prueba de asignar valor a variable con la Salida de un Comando
time**=**`date| cut -c12-19`
echo LA HORA ES : $time

$t

LA HORA ES : 11:03:32

4.2.3.4. Asignación de parámetros posicionales.

Como valor de la variable se almacena el valor de un parámetro posicional. El formato es:

var**=$**número

Ejemplo:

$cat simp.p

#Prueba de asignar a una variable un Parametro posicional var**=$**1
echo El valor de la variable es : $var

$simp.p **Hola**

El valor de la variable es : Hola

4.2.4. HERE DOCUMENT

Permite situar dentro de un programa shell, líneas que son redireccionadas para ser la entrada de un comando. En otras palabras permite colocar todos los subcomandos de una utilidad dentro del mismo shell sin necesidad de usar un archivo aparte para ello. La notación consiste del símbolo **<<** y un delimitador que especifica el comienzo y el final de las líneas que se tomaran como entrada del comando. En nuestro ejemplo usaremos como delimitador el símbolo **!**. El formato es :

comando **<<delimitador**
  linea1
  linea2
  ...
delimitador

Ejemplo: Es muy conveniente para usar el editor **ed** dentro de un programa shell. Supongamos que se desea hacer un programa que use el editor ed, haga una sustitución global de un texto en el archivo, escriba estos cambios y salga del editor. Observe que el signo **-**usado con el ed, previene el conteo de caracteres.

$cat cambiar.pmga

```
#Prueba uso de Here Document,para hacer sustitución
#global de una palabra en un archivo.
echo  Digite el nombre del archivo:
read arch1
echo  Digite texto a ser cambiado:
read textov
echo  Digite nuevo texto :
read texton
ed - $arch1 <<!
g/$textov/s//$texton/g
w
q
!
```

4.2.5. COMANDO TEST

El comando **test** se usa muy frecuentemente junto con el **if** y con las condicionales simples, ya que sirve para probar muchas condiciones.

Su formato general es

test expresión ó **[expresión]**

y su estado de salida es cero ("true") si la expresión es cierta y diferente de cero ("false") en caso contrario. (El comando **test** tiene un "link" que es el comando **[**. Pero si se invoca en esta forma, debe colocarse a continuación de la expresión un **]**. Esto existe sólo por razones "estéticas").

Las expresiones pueden ser:

4.2.5.1. Manejo de "strings".

string1 **=** string2 que retorna "true" si los dos strings son idénticos.

string1 **!=** string2 que retorna "true" silos dos strings no son idénticos.

string retorna "true" si el string no es nulo (vacío)

-n string "true" no es nulo.

-z string "true" si es nulo.

Por ejemplo:

$test $HOME **=** `pwd` && echo estamos en el directorio "home"

Note que se están comparando dos tiras de caracteres: el contenido de la variable HOME (que siempre existe y dice cuál es el directorio "home" y el resultado del comando **pwd**). Si la comparación es exitosa, se ejecutará el comando **echo**.

4.2.5.2. Operadores para enteros.

-eq Igual
-ge Mayor o igual
-gt Mayor que
-le Menor o igual
-lt Menor que
-ne Diferente

Por ejemplo:

$[40 **-lt** `ls | wc -l`] && echo Hay muchos archivos aquí

Aquí, el **test** (ó **[**) está comparando 40 con el resultado del "pipeline" **ls** | **wc -l** que retorna el número de archivos en el directorio corriente. Note que (como en el

ejemplo anterior) se está usando la "sustitución de comandos" (`**...**`) que produce como resultado la salida estándar de los comandos ejecutados.

4.2.5.3. Operadores de archivos.

-d El archivo es un directorio
-f El archivo es ordinario
-r El archivo es legible
-s El archivo tiene longitud mayor de 0
-w Se tiene permiso de escritura en el archivo
-x El archivo es ejecutable

Por ejemplo:

$ test **-r** /etc/motd && **echo "El archivo es legible"**

Los operadores anteriores pueden además mezclarse con los operadores booleanos:

! Not
-a And (y lógico)
-o Or (o lógico)

Por ejemplo:

$test $CONTADOR **-ge** 0 **-a** $CONTADOR **-lt** 10

Resultará cierto si el contenido de la variable CONTADOR está entre 0 y 10.

4.2.6. PROPOSICION FOR

Esta estructura permite ejecutar un grupo de instrucciones para cada una de las palabras incluidas en una lista. Su formato general es:

for variable **in** palabra_1 palabra_2 ...
do
        comandos
done

Los "comandos" serán ejecutados una vez por cada una de las palabras de la lista (palabra_1, palabra_, ...). La variable $variable será cargada con la palabra corriente.

Ejemplo:

$ cat for.pgma

#Prueba de comando for
#Este programa listara el numero de líneas de archivos #que empiecen por la letra s
for prog **in** s*
do
echo $prog tiene `wc -l $prog` líneas
done

Si verificamos que existen archivos que empiecen con la letra **s** encontraremos:

$ ls -x s*

salida salidaat shell.sh show.param simp.p

Ejecutamos el programa y obtenemos:

$ for.pgma

salida tiene 9 salida líneas
salidaat tiene 43 salidaat líneas
shell.sh tiene 4 shell.sh líneas
show.param tiene 3 show.param líneas
simp.p tiene 4 simp.p líneas

El **for** también puede usarse así:

for palabra
    comandos
done

En este caso palabra tomas sucesivamente cada uno de los parámetros posicionales con que se invocó el comando.

4.2.7. PROPOSICIONES WHILE Y UNTIL

Estas dos proposiciones permiten ejecutar repetitivamente un grupo de comandos.

Sintaxis general de la estructura **while**:

while condición
do
    lista_de_comandos
done

La lista de comandos se ejecutará mientras el comando "condición" de un estado de salida "true".

La sintaxis general de la estructura **until** es:

until condición
    lista_de_comandos
done

La lista de comandos se ejecutará mientras el programa "condición" de un estado de salida "false".

Es frecuente usar como "condición" el comando **test**.

Ejemplos:

$cat while.pgma

```
#Prueba del comando While
#Este programa leerá nombres y los enviara a un archivo
#llamado nombres hasta que digiten "nada"
name="temp"
while test $name != "nada"
do
  echo Digite el nombre :
  read name
  echo $name >>nombres
done
```

$cat until.pgma

```
#Prueba del comando Until
#Este programa leerá nombres y los enviara a un archivo
#nombres hasta que digiten "nada"
name="temp"
until test $name = "nada"
do
  echo Digite el nombre :
  read name
  echo $name >>nombres
done
```

4.2.8. CONDICIONAL IF

La forma general de este condicional es:

```
if condición
then
        comando ...
else
        comando ...
```

fi

Las palabras **if, then, else** y **fi** son palabras clave, que deben escribirse tal cual. La **condición** es algún comando del UNIX que retorne un estado de salida "true" o "false". Los comando son cualquier programa ejecutable del UNIX.

La parte del **else** es opcional.

Se pueden usar proposiciones **if** anidadas, así:

```
if condición
then
        comandos
elif condición
        comandos
else
        comandos
fi
```

Por ejemplo:

$ cat busca.pgma

```
#Prueba de comando If
#Este programa busca una palabra en un archivo e indica
#si la encontró  o no
echo  Digite la palabra a buscar y el archivo:
read palabra archivo
if
 grep $palabra $archivo
then
 echo LA PALABRA $palabra ESTA EN EL ARCHIVO $archivo
else
 echo  LA PALABRA $palabra NO ESTA EN EL ARCHIVO $archivo  fi
```

Verificamos el archivo en el cual vamos a buscar :

$ cat salida

/usr/people/hector/cursoshell
Wed Feb 17 10:10:57 EST 1993
total 7
-rw-r--r-- 1 root other 0 Feb 17 10:10 batch_at
-rw-r--r-- 1 root other 317 Feb 17 10:08 listado
-rwxrwxrwx 1 root other 413 Feb 17 10:07 menu
-rw-r--r-- 1 hector other 1207 Feb 17 09:11 metacarac
-rwxrwxrwx 1 root other 52 Feb 17 09:47 prueba.bat -rw-r--r-- 1 root other 59 Feb 17 10:10 salida

Ejecutamos el programa :

$ busca.pgma

Digite la palabra a buscar y el archivo:
hector salida

/usr/people/hector/cursoshell
-rw-r--r-- 1 hector other 1207 Feb 17 09:11 metacarac

LA PALABRA hector ESTA EN EL ARCHIVO salida

$ busca.pgma

Digite la palabra a buscar y el archivo:
carlos salida

LA PALABRA carlos NO ESTA EN EL ARCHIVO salida

4.2.9. SALIDA A /DEV/NULL

El sistema de archivos tiene un archivo llamado /**dev**/**null** donde se pueden depositar salidas no deseadas. En el ejemplo anterior, el comando grep al encontrar una palabra en un archivo, despliega en la pantalla las líneas que la

contienen. Para nuestro caso que solo deseábamos verificar si existía o no, podemos no querer que se presente por pantalla estas líneas. Estas salidas se pueden direccionar al archivo /dev/null. Ejemplo:

Modificamos el programa anterior :

$cat ifnull.pgma

```
#Prueba de  salida a /dev/null
#Este programa busca una palabra en un archivo e indica
#si la encontró o no
echo  Digite la palabra a buscar y el archivo:
read palabra archivo
if
 grep $palabra $archivo >/dev/null
then
 echo LA PALABRA $palabra ESTA EN EL ARCHIVO $archivo
else
 echo  LA PALABRA $palabra NO ESTA EN EL ARCHIVO $archivo
fi
```

$ ifnull.pgma

Digite la palabra a buscar y el archivo:
hector salida

LA PALABRA hector ESTA EN EL ARCHIVO salida

4.2.10. COMANDOS BREAK Y CONTINUE

El comando **break** detiene incondicionalmente la ejecución de cualquier loop, en el cual este se encuentre y continua la ejecución después de alguna de las instrucciones **done**, **fi** o **esac**. Si no hay comandos después de estas instrucciones, el programa termina.

El comando **continue** hace que el programa vaya a la siguiente iteración del ciclo , sin ejecutar el resto de instrucciones en el loop.

4.2.11. PROPOSICION CASE

Esta proposición se usa para comparar el contenido de una variable contra una serie de valores y ejecutar una lista de comandos de acuerdo con el valor que contiene la variable (caso típico: un menú).

El formato general es:

```
case palabra in
        patrón1)        comando
                        ...
                        comando;;
        patrón2)        comando
                        ...
                        comando;;
           ...)
                        ...;;
esac
```

El contenido de la variable "palabra" es comparado contra cada uno de los patrones (patrón1, patrón2, ...) hasta que coincida con alguno. En eses momento, se ejecutan todos los comandos a continuación del patrón, hasta que se encuentren los dos punto y comas. Si la "palabra" no coincide con ninguno de los patrones, no se ejecuta ninguno de los comandos dentro de **case**.

Los patrones pueden tener los metacaracteres ***, []** y **?**, con el significado usual. También se pueden usar, dentro de los patrones, varias palabras separadas por barras verticales, para indicar "una cualquiera de ellas" (por ejemplo **a**|**A**).

Ejemplo: Se presentará un menú simple, que realiza ciertas tareas sencillas.

```
$ cat case.pgma
```

```
while  true
do
  echo          "     "
  echo "***********************************"
  echo  1.  VER USUARIOS DENTRO DEL SISTEMA
  echo  2.  VER ESTADO DEL SPOOL
  echo  3.  VER PROCESOS DEL SISTEMA
  echo  4.  TERMINAR MENU
  echo "***********************************"
  echo "    "
  echo Digite opcion:
  read opt
  case $opt  in
     1)   who;;
     2)   lpstat -t;;
     3)   ps -ef;;
     4)   break;;
     *)   echo "Opcion invalida, intente de nuevo";;
  esac
done
```

```
$case.pgma

**********************************
1. VER USUARIOS DENTRO DEL SISTEMA
2. VER ESTADO DEL SPOOL
3. VER PROCESOS DEL SISTEMA
4. TERMINAR MENU
**********************************

Digite opcion:
1

oracle     ttyq0        Feb 17 08:23
root       tty0         Feb 17 09:58
hector     ttyq2        Feb 17 13:36

**********************************
1. VER USUARIOS DENTRO DEL SISTEMA
2. VER ESTADO DEL SPOOL
```

```
3. VER PROCESOS DEL SISTEMA
4. TERMINAR MENU
**********************************

Digite opcion:
4

$
```

4.2.12. DEPURACION DE PROGRAMAS SHELL

El lenguaje de programación de la shell tiene los mismos problemas de depuración de cualquier lenguaje de programación. Muchas veces se pueden tener errores de lógica.

La mejor forma de seguir el funcionamiento de un programa de la shell es invocarlo usando la bandera **-x** que hace que cada comando que va ser ejecutado, sea mostrado previamente en la pantalla.

Suponga que se tiene por ejemplo un "script" llamado **lsdir** cuyo contenido es:

$cat lsdir

```
if [ $# = 0 ]
then
        dir=.
else
        dir=$1
fi
find $dir -type d -print
```

Este "script" mostrará el directorio dado como parámetro, si es que es directorio.

Al ejecutarlo así se producirá una salida como la siguiente:

$**sh -x** lsdir

```
+[ 0 = 0]
dir=.
+find . -type d - print
.
./fuentes
./bin
./docs
```

Observe que cada comando invocado por la shell va precedido de un signo **+**.

4.2.13. COMANDO EXPR

La shell no tiene facilidades para calcular expresiones aritméticas. Cuando ellas son necesarias, se usa el comando **expr**. Este comando espera como parámetros los elementos de una expresión aritmética y su resultado es escrito por la unidad de salida estándar. La forma general de los parámetros es:

expresión **operador** expresión

Los operadores pueden ser los siguientes (se muestran de menor a mayor precedencia):

\& Retorna el operando de la izquierda si ambos operandos son no nulos y diferentes de cero En caso contrario, retorna cero (Note que es necesario escapar con \ el **&** porque de otra forma sería interpretado como ejecución en background por la shell.

\| Retorna el operando de la izquierda si es cero o nulo. En caso contrario retorna el de la derecha (Note que es necesario escapar con \ el | porque de otra forma sería interpretado como "pipe" por la shell.

!= < <= = > >=

Estos operadores realizan comparaciones. Si ambos operadores son enteros, se realiza una comparación numérica. Si no, se realiza una

comparación lexicográfica. Si la comparación es cierta, **expr** retorna 1 y si no, cero.

+ - * / %

Retorna la suma, la resta, La multiplicación, división o módulo, respectivamente.

: Este es un operador para comparación de caracteres. El operador de la izquierda es un "string" cualquiera y el de la derecha, una expresión regular (vea 2.1.1.). El resultado es el número de caracteres del primer operando, que coincidieron con la expresión regular. Sin embargo si parte de la expresión regular está encerrada entre **\(...\)** el resultado será los caracteres que coincidieron.

Ejemplos.

$**expr \(** 5 * 9 **\) %** 4 imprimirá 1

La siguiente secuencia de comandos:

```
$contador=4
$while [ $contador -gt 0 ]
>do
>echo $contador
>contador = `expr $contador - 1`
>done
```

...imprimirá

```
4
3
2
1
```

Un ejemplo del uso de **expr** para comparar con expresiones regulares, sería el siguiente: supongamos que se desea mostrar el nombre de la terminal en la que el usuario está trabajando. Esto se logra con el comando **tty**:

$tty
/dev/tty12

Pero supongamos, que se desea suprimir el /dev/. Podría lograrse con:

$**expr** `tty` **:** "/dev/**(**.***)**"
tty12

Al extraer la parte de la expresión regular que coincide con **.*** después de /**dev**/, se estará mostrando sólo el nombre de la terminal.

4.3. EJERCICIO

Realizar un programa para presentar un menú, que le permita al usuario :

- Verificar si un usuario existe en el sistema.
- Invocar el editor visual.
- Consultar el número de usuarios.

Desarrollo:

```
#
#       Ejemplo de menú
#       Note (de paso) que el carácter # sirve para iniciar
#       un comentario
while true    # true es un programa que siempre retorna 0
do
clear
echo "          MENÚ  PRINCIPAL"
echo " "
echo "   V. Verificar si usuario existe"
echo "   I. Invocar editor vi"
echo "   C. Consultar numero de usuarios"
echo "   S. Salir de este shell"
echo " "
```

```
echo "   Digite su opcion \c"

read op
case $op in
        V|v)
                clear
                echo "DE EL NOMBRE DE USUARIO \c"
                read NOM
                grep $NOM /etc/passwd >/tmp/arch$$
                if [ -s /tmp/arch$$ ]
                then
                        echo "El nombre completo del usuario $NOM es `cut -f 5 -d:
                        /tmp/arch$$`"
                else
                        echo "El usuario $NOM no existe"
                fi
                rm /tmp/arch$$;; #Borrar el archivo de trabajo

        E|e)
                clear
                echo "De el nombre del archivo \c"
                read arch
                vi $arch;;
        C|c)
                clear
                echo "El numero de usuarios es `who |wc -l`.";;
        S|s)
                exit 0 ;;
        *)
                echo "Opcion invalida \007" ;;
esac
echo "Presione return para continuar"
read t
done
```

4.4. TALLER EN CLASE

Realizar un menú para el manejo de impresión del sistema que presente las siguientes opciones:

```
                        MANEJO DE IMPRESIÓN

1. Ver Estado del sistema de impresion
2. Ver impresora por defecto
3. Ver impresoras y estado
4. Ver cola de trabajo de una impresora
5. Deshabilitar la impresion por una impresora
6. Permitir la impresion por una impresora
7. Permitir ingreso de trabajos a la cola de una impresora
8. NO permitir ingreso de trabajos a la cola de una impresora
9. Cancelar un trabajo de una impresora
10. Mover cola de trabajo de una impresora a otra
11. Reenviar un trabajo de impresion desde una pagina
12. Activar sistema de spool
13. Desactivar sistema de spool
14. salir del menu

 Digite la opcion deseada:
```

En todas las opciones se deben hacer las validaciones respectivas (que la impresora escogida exista, que se digite algo valido, etc).

4.5. DESARROLLO DEL TALLER DE CLASE.

El archivo llamado menuimp, que contiene el código de programación shell para el anterior ejercicio es:

```
while true
do
    clear
    echo "  MANEJO DE IMPRESION "
    echo "1. Ver Estado del sistema de impresion"
    echo "2. Ver impresora por defecto "
    echo "3. Ver impresoras y estado"
    echo "4. Ver cola de trabajo de una impresora"
    echo "5. Deshabilitar la impresion por una impresora"
    echo "6. Permitir la impresion por una impresora"
    echo "7. Permitir ingreso de trabajos a la cola de una impresora"
    echo "8. NO permitir ingreso de trabajos a la cola de una impresora"
    echo "9. Cancelar un trabajo de una impresora"
    echo "10. Mover cola de trabajo de una impresora a otra"
    echo "11. Reenviar un trabajo de impresion desde una pagina "
    echo "12. Activar sistema de spool "
    echo "13. Desactivar sistema de spool "
    echo "14. salir del menu"
```

```
    echo " "
    echo " Digite la opcion deseada: "
    read opc
    clear
    imp=""
    case $opc in
      1 ) lpstat -r | grep not > /dev/null
          if [ $? -eq 0 ]
            then
                banner " DESACTIVADO"
            else
                banner " ACTIVADO"
          fi;;
      2) lpstat -d | grep "^no system" > /dev/null
          if [ $? -eq 0 ]
            then
                banner "NO HAY" DEFINIDA
            else
                banner "`lpstat -d |cut -f 2 -d:`"
          fi;;
      3) cd /var/spool/lp/admins/lp/interfaces
         for imp in *
         do
            echo " INFORMACION DE IMPRESORA $imp"
            echo " ----------------------------------"
            lpstat -t |grep "^$imp not accepting" >/dev/null && echo "NO
PERMITE TRABAJOS EN LA COLA DE IMPRESION"
            lpstat -t |grep "^$imp accepting" >/dev/null && echo "PERMITE
TRABAJOS EN LA COLA DE IMPRESION"
            lpstat -t |grep "$imp is idle" >/dev/null && echo
"ACTUALMENTE ESTA EN LINEA PERO OCIOSA"
            lpstat -t |grep "$imp disabled" >/dev/null && echo
"ACTUALMENTE ESTA FUERA DE LINEA"
            echo " ----------------------------------"
            echo " "
         done;;
      4) rm /tmp/parte2 /tmp/parte1 /tmp/numeros 2> /dev/null
         rm /shell/guia/* 2> /dev/null
         echo "Digite el nombre de la impresora"
         read imp
         test -n "$imp" && impre=$imp
         while [ -z  "$impre" ]
           do
              echo " NO HA DIGITADO NADA , POR FAVOR DIGITE EL NOMBRE DE
LA IMPRESORA"
              read impre
           done
         ls /var/spool/lp/admins/lp/interfaces/$impre 2>/dev/null
>/dev/null
         if [ $? -ne 0 ]
           then
              echo " LA IMPRESORA $impre NO EXISTE EN EL SISTEMA "
```

```
              read algo
              continue
         fi
         lpstat -o | grep "$impre" > /dev/null
         if [ $? -ne 0 ]
           then
              echo " COLA DE LA IMPRESORA $impre ESTA VACIA "
              break
         fi
         echo " COLA DE IMPRESION  DE $impre"
         echo " ------------------------------------------------------
------"
         echo " No.         Usuario       Tamano            Nombre"
         echo " peticion                          archivo       archivo"
         echo " ------------------------------------------------------
------"
         lpstat -o | grep "^$impre" | cut -f2 -d"-"|cut -c1-43
>/tmp/parte1
         lpstat -o |grep "^$impre" |cut -f2 -d"-" | cut -f1 -d" "
>/tmp/numeros
         bash /shell/pruebita
         bash /tmp/numeros
         rm /tmp/parte2 2> /dev/null
         touch /tmp/parte2
         cd /shell/guia
         for num in *
          do
            cd /var/spool/lp/tmp/cslb50

            grep "^F" $num |cut -f2 -d" " >> /tmp/parte2
          done
        paste /tmp/parte1 /tmp/parte2 ;;
      5)  echo "Digite el nombre de la impresora"
         read imp
         test -n "$imp" && impre=$imp
         while [ -z  "$impre" ]
           do
              echo " NO HA DIGITADO NADA , POR FAVOR DIGITE EL NOMBRE DE
LA IMPRESORA"
              read impre
           done
         ls /var/spool/lp/admins/lp/interfaces/$impre 2>/dev/null
>/dev/null
         if [ $? -ne 0 ]
           then
              echo " LA IMPRESORA $impre NO EXISTE EN EL SISTEMA "
              read algo
              continue
         fi
         /usr/bin/disable $impre >/dev/null && banner IMPRESORA $impre
DESHABILITADA;;
      6) echo "Digite el nombre de la impresora"
```

```
         read imp
         test -n "$imp" && impre=$imp
         while [ -z  "$impre" ]
           do
              echo " NO HA DIGITADO NADA , POR FAVOR DIGITE EL NOMBRE DE
LA IMPRESORA"
              read impre
           done
         ls /var/spool/lp/admins/lp/interfaces/$impre 2>/dev/null
>/dev/null
         if [ $? -ne 0 ]
           then
              echo " LA IMPRESORA $impre NO EXISTE EN EL SISTEMA "
              read algo
              continue
         fi
         /usr/bin/enable $impre >/dev/null && banner IMPRESORA $impre
HABILITADA;;
      7) echo "Digite el nombre de la impresora"
         read imp
         test -n "$imp" && impre=$imp
         while [ -z  "$impre" ]
           do
              echo " NO HA DIGITADO NADA , POR FAVOR DIGITE EL NOMBRE DE
LA IMPRESORA"
              read impre
           done
         ls /var/spool/lp/admins/lp/interfaces/$impre 2>/dev/null
>/dev/null
         if [ $? -ne 0 ]
           then
              echo " LA IMPRESORA $impre NO EXISTE EN EL SISTEMA "
              read algo
              continue
         fi
         /usr/lib/accept $impre >/dev/null && banner IMPRESORA $impre
ACEPTA;;
      8) echo "Digite el nombre de la impresora"
         read imp
         test -n "$imp" && impre=$imp
         while [ -z  "$impre" ]
           do
              echo " NO HA DIGITADO NADA , POR FAVOR DIGITE EL NOMBRE DE
LA IMPRESORA"
              read impre
           done
         ls /var/spool/lp/admins/lp/interfaces/$impre 2>/dev/null
>/dev/null
         if [ $? -ne 0 ]
           then
              echo " LA IMPRESORA $impre NO EXISTE EN EL SISTEMA "
              read algo
```

```
             continue
          fi
          /usr/lib/reject $impre >/dev/null && banner IMPRESORA $impre
RECHAZA;;
       9) rm /tmp/parte2 /tmp/parte1 /tmp/numeros 2> /dev/null
          rm /shell/guia/* 2> /dev/null
          echo "Digite el nombre de la impresora"
          read imp
          test -n "$imp" && impre=$imp
          while [ -z  "$impre" ]
            do
               echo " NO HA DIGITADO NADA , POR FAVOR DIGITE EL NOMBRE DE
LA IMPRESORA"
               read impre
            done
          ls /var/spool/lp/admins/lp/interfaces/$impre 2>/dev/null
>/dev/null
          if [ $? -ne 0 ]
            then
               echo " LA IMPRESORA $impre NO EXISTE EN EL SISTEMA "
               read algo
               continue
          fi
          lpstat -o | grep "$impre" > /dev/null
          if [ $? -ne 0 ]
            then
               echo " COLA DE LA IMPRESORA $impre ESTA VACIA "
               break
          fi
          echo " COLA DE IMPRESION  DE $impre"
          echo " ------------------------------------------------------
------"
          echo " No.         Usuario       Tamano            Nombre"
          echo " peticion                          archivo       archivo"
          echo " ------------------------------------------------------
------"
          lpstat -o | grep "^$impre" | cut -f2 -d"-"|cut -c1-43
>/tmp/parte1
          lpstat -o |grep "^$impre" |cut -f2 -d"-" | cut -f1 -d" "
>/tmp/numeros
          bash /shell/pruebita
          bash /tmp/numeros
          rm /tmp/parte2 2> /dev/null
          touch /tmp/parte2
          cd /shell/guia
          for num in *
           do
             cd /var/spool/lp/tmp/cslb50

             grep "^F" $num |cut -f2 -d" " >> /tmp/parte2
           done
         paste /tmp/parte1 /tmp/parte2
```

```
        echo "-------------------------------------------------------------
----"
        echo " DIGITE EL NUMERO DE LA PETICION QUE DESEA CANCELAR : "
        pet=""
        read p
         test -n "$p" && pet=$p
         while [ -z  "$pet" ]
           do
              echo " NO HA DIGITADO NADA , POR FAVOR DIGITE EL NUMERO DE
LA PETICION"
              read pet
           done
         cancel $impre-$pet >/dev/null 2> /dev/null
         if [ $? -ne 0 ]
           then
              echo " EL NUMERO DE PETICION NO EXISTE "
              continue
         fi
         echo " EL TRABAJO $impre-$pet FUE CANCELADO EXITOSAMENTE" ;;
      10) echo "Digite el nombre de la impresora origen , la cual se
desactivara automaticamente"
         read imp
         test -n "$imp" && impre=$imp
         while [ -z  "$impre" ]
           do
              echo " NO HA DIGITADO NADA , POR FAVOR DIGITE EL NOMBRE DE
LA IMPRESORA"
              read impre
           done
         ls /var/spool/lp/admins/lp/interfaces/$impre 2>/dev/null
>/dev/null
         if [ $? -ne 0 ]
           then
              echo " LA IMPRESORA $impre NO EXISTE EN EL SISTEMA "
              read algo
              continue
         fi
         lpstat -o | grep "$impre" > /dev/null
         if [ $? -ne 0 ]
           then
              echo " COLA DE LA IMPRESORA $impre ESTA VACIA , NO SE
JUSTIFICA MOVER A OTRA COLA"
              break
         fi
         echo "Digite el nombre de la impresora destino , la cual se debe
estar aceptando trabajos"
         read imp
         test -n "$imp" && impre2=$imp
         while [ -z  "$impre2" ]
           do
              echo " NO HA DIGITADO NADA , POR FAVOR DIGITE EL NOMBRE DE
LA IMPRESORA"
```

```
             read impre2
           done
         ls /var/spool/lp/admins/lp/interfaces/$impre2 2>/dev/null
>/dev/null
         if [ $? -ne 0 ]
           then
             echo " LA IMPRESORA $impre2 NO EXISTE EN EL SISTEMA "
             read algo
             continue
         fi
        lpstat -t |grep "^$impre2 not accepting" >/dev/null
         if [ $? -eq 0 ]
           then
             echo " $impre2 NO PERMITE TRABAJOS EN LA COLA DE IMPRESION"
             continue
         fi
         /usr/lib/reject >/dev/null 2>/dev/null $impre && echo " LA
IMPRESORA $impre NO RECIBIRA MAS TRABAJOS "
         /usr/bin/disable >/dev/null 2>/dev/null $impre && echo " LA
IMPRESORA $impre NO IMPRIMIRA MAS TRABAJOS "
         lpmove $impre $impre2 >/dev/null 2>/dev/null && echo " COLA DE
IMPRESION DE $impre MOVIDA EXITOSAMENTE A $impre2 " ;;
      11) rm /tmp/parte2 /tmp/parte1 /tmp/numeros 2> /dev/null
         rm /shell/guia/* 2> /dev/null
         echo " Esta opcion permite reenviar por la misma impresora , un
trabajo que esta en cola, pero desde una pagina determinada"
         echo "Digite el nombre de la impresora donde reside el trabajo,
la cual se deshabilitara momentaneamente"
         read imp
         test -n "$imp" && impre=$imp
         while [ -z  "$impre" ]
           do
             echo " NO HA DIGITADO NADA , POR FAVOR DIGITE EL NOMBRE DE
LA IMPRESORA"
             read impre
           done
         ls /var/spool/lp/admins/lp/interfaces/$impre 2>/dev/null
>/dev/null
         if [ $? -ne 0 ]
           then
             echo " LA IMPRESORA $impre NO EXISTE EN EL SISTEMA "
             read algo
             continue
         fi
         lpstat -o | grep "$impre" > /dev/null
         if [ $? -ne 0 ]
           then
             echo " COLA DE LA IMPRESORA $impre ESTA VACIA , NO HAY
NINGUN TRABAJO PARA REENVIAR "
             break
         fi
         echo " COLA DE IMPRESION  DE $impre"
```

```
         echo " ------------------------------------------------------------
------"
         echo " No.         Usuario       Tamano            Nombre"
         echo " peticion                          archivo       archivo"
         echo " ------------------------------------------------------------
------"
         lpstat -o | grep "^$impre" | cut -f2 -d"-"|cut -c1-43
>/tmp/parte1
         lpstat -o |grep "^$impre" |cut -f2 -d"-" | cut -f1 -d" "
>/tmp/numeros
         bash /shell/pruebita
         bash /tmp/numeros
         rm /tmp/parte2 2> /dev/null
         touch /tmp/parte2
         cd /shell/guia
         for num in *
          do
            cd /var/spool/lp/tmp/cslb50

            grep "^F" $num |cut -f2 -d" " >> /tmp/parte2
          done
        paste /tmp/parte1 /tmp/parte2
        echo "------------------------------------------------------------
----"
        echo " AL DIGITAR EL NUMERO DE LA PETICION Y EL NUMERO DE PAGINA
DESDE DONDE DESEA ENVIAR"
        echo " LA PETICION SE ENVIARA DE NUEVO DESDE ESA PAGINA Y LA
ANTIGUA PETICION SE CANCELARA"
        echo " DEBE ACTIVAR LA IMPRESORA DE NUEVO , UNA VEZ ESTE SEGURO "
        echo " DIGITE EL NUMERO DE LA PETICION QUE DESEA REENVIAR  : "
        read p
         test -n "$p" && pet=$p
         while [ -z  "$pet" ]
           do
              echo " NO HA DIGITADO NADA , POR FAVOR DIGITE EL NUMERO DE
LA PETICION"
              read pet
           done
         echo " DIGITE EL NUMERO DE PAGINA DESDE DONDE DESEA ENVIAR EL
TRABAJO Y EL SIZE DE LA HOJA DE IMPRESION ( EN LINEAS)"
         echo " DIGITE EL SIZE DE LA PAGINA EN LINEAS ( EJM: 50 LINEAS,
NO SE VALIDARA LO LEIDO) : "
         read lineas
         echo " DIGITE EL NUMERO DE LA PAGINA DESDE DONDE DESEA ENVIAR EL
ARCHIVO DE SU PETICION ( NO SE VALIDARA): "
         read pagina
         archivo=`grep "^F" /var/spool/lp/tmp/cslb50/$pet-0 |cut -f2 -d"
" `
         echo $archivo
         read algo
         pr -l $lineas +$pagina $archivo | lp -d$impre >/dev/null
         if [ $? -eq 0 ]
```

```
           then
              cancel $impre-$pet >/dev/null 2> /dev/null && echo "
PETICION ANTERIOR $pet FUE CANCELADA"
              echo "el archivo $archivo de la peticion $pet , fue enviado
de nuevo desde la pagina $pagina"
              continue
          fi
          echo " FALLO EL REENVIO DEL ARCHIVO $archivo DE LA PETICION
$pet" ;;
      12) /usr/lib/lpsched && banner SPOOL ACTIVADO ;;
      13) /usr/lib/lpshut && banner SPOOL APAGADO ;;
      14) exit;;
      *) echo " OPCION INVALIDA, DIGITE DE NUEVO" ;;
    esac
    read algo
  done
```

Existen algunas consideraciones importantes , que debemos tener en cuenta en este tipo de menus:

- Generalemente se corren como superusuario o con un usuario con privilegios, así que no es conveniente que se permita al usuario , salir del menú y quedar en línea de comandos.
- Se deben realizar todas las validaciones posibles para que sean menús que permitan a usuarios sin conocimientos avanzados operar las máquinas.

4.6. MANEJO DE LAS INTERRUPCIONES EN UNIX

Las señales son enviadas a un proceso para informarle que un evento a ocurrido. Un proceso puede manejar o ignorar la señal. La acción por defecto para la mayoría de las señales es terminación del proceso. El único que puede enviar señales a cualquier proceso es el superusuario.

Ejemplos de señales comunes:

EXIT (0): Usada para ejecutar actividades de logout, ej, limpiar pantalla.

HUP (1): Enviada a todos los procesos de una sesión cuando el shell termina. Termina ademas todos los procesos corriendo en background , a no ser que hayan sido lanzados con nohup.

INT (2): Enviada cuando se precisona el caracter de interrupción (definido con stty intr char). Por defecto es CTRL.-C.

QUIT (3): Enviada cuando se precisona el caracter de quit (definido con stty quit char). Por defecto es ^\.

TERM (15): Terminación de software. Señal default del comando kill.

KILL (9): Kill forzoso. No puede ser manejada o ignorada.

Para manejar o ignorat las interrupciones (señales) se utiliza el comando trap.

trap “lista de comandos “ lista de señales

Ejm:

trap “echo presiono interrupcion 2 “ INT
trap “echo presiono interrupcion 3 “ QUIT

trap “ “ lista de señales

Ejm: Deshabilitar las señales

trap “” INT QUIT EXIT

Ejm: En el siguiente menu:

```
# cat menu1

while true
do
  echo " menu "
  echo " 1. ver usuarios trabajando "
  echo " 2. ver procesos actuales "
  echo " 3. salir"
  echo " digite su opcion: "
  read opcion
  case $opcion in
    1)  who |more ;;
    2)  ps -ef |more ;;
    3)  exit ;;
    *)  echo " opcion invalida";;
```

```
  esac
done
```

El usuario al presionar las teclas que causan la interrupción INT, puede salir del menu y quedar en línea de comandos. La combinación de teclas de esa interrupción es:

```
# stty -a
speed 38400 baud; rows 20; columns 100; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>; eol2 = <undef>; start = ^Q;
stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 -hupcl -cstopb cread -clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -ixoff -iuclc -ixany
-imaxbel
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt echoctl echoke
```

Al correr el menu y presionar ctrl.-c se genera una señal intr. y causa que el progama termine. Para que el programa shell maneje las señales provenientes de : INT, QUIT, TERM y STOP. Al presionar alguna de las secuencias de teclas que las generan el programa envia un mensaje a la pantalla.

```
trap " echo SECUENCIA INVALIDA" 2 3 17 15
while true
do
  echo " menu "
  echo " 1. ver usuarios trabajando "
  echo " 2. ver procesos actuales "
  echo " 3. salir"
  echo " digite su opcion: "
  read opcion
  case $opcion in
    1)  who |more ;;
    2)  ps -ef |more ;;
    3)  exit ;;
    *)  echo " opcion invalida";;
  esac
done
```

Al correr:

```
# menu1
 menu
 1. ver usuarios trabajando
 2. ver procesos actuales
 3. salir
```

```
 digite su opcion:
SECUENCIA INVALIDA
```

Ejercicio :

En el menú de manejo del spool, manejar las señales más comunes, mostrando un mensaje sobre que tipo de señal es.

4.7. UTILIDAD AWK

Es un filtro potente de unix. No voy a presentar toda su sintaxis , pues se consulta en la página de man respectiva. Mostraré algunos ejemplos para ver algunas de sus características.

Ejm: Archivo con nombres de personas, sexo (M/F), edad , telefono y si fuman (S/N).

```
# cat nombres

hector M 38 6369388 N
diana F 9 6313142 N
patricia F 35 550909 S
carlos M 42 765438 S
```

Los registros y campos se numeran diferente en awk. Por ejemplo $1 es el primer campo, $2 el segundo, etc. $0 es el registro completo.

Las órdenes awk se definen en función de patrones y acciones. Estos patrones y acciones se pueden invocar desde línea de comandos o leerlos desde un archivo. Si invoco el comando

```
# awk '{ print $0 } ' nombres

hector M 38 6369388 N
diana F 9 6313142 N
patricia F 35 550909 S
carlos M 42 765438 S
```

Estoy indicándole al awk que procese todas las líneas del archivo nombres y que muestre en pantalla todo el registro.

```
# awk '{ print $1 , $4 } ' nombres

hector 6369388
diana 6313142
patricia 550909
carlos 765438
```

Estoy indicándole al awk que procese todas las líneas del archivo nombres y que muestre en pantalla sólo los campos $1 (el nombre) y $4 (el teléfono).

Si tengo un archivo con las órdenes del awk, donde le digo que el separador de campos al leer del archivo es el espacio y que al salir el separador de campos es 3 TAB, y lo ejecuto:

```
# cat ordenes
{
FS = " "
OFS = "                 "
}
{print $1 , $4}
```

```
# awk -f ordenes nombres
hector                  6369388
diana                   6313142
patricia                 550909
carlos                  765438
```

Ahora le voy a indicar al archivo de órdenes que solo muestre (print) los campos uno y cuatro de los registros, cuyo segeundo campo tengan una letra M (masculino).

```
# cat ordenes
{
FS = " "
OFS = "              "
}
```

```
($2 == "M") {print $1 , $4}

# awk -f ordenes nombres
hector              6369388
carlos                765438
```

Otras acciones:

```
# cat ordenes
{
FS = " "
OFS = "                "
}
($2 == "M") && ($5 == "S") {print $1 , $4}
```

Mostrar los nombres y el telefono de los hombres que fuman.

```
# awk -f ordenes nombres

carlos                765438
```

Incluí en el archivo nombres un indicativo en el teléfono

```
# cat nombres
hector M 38 (7)6369388 N
diana F 9 (4)6313142 N
patricia F 35 (4)550909 S
carlos M 42 (3)765438 S
maria F 18 (3)7654356 S
```

Solo deseo ver los nombres de las personas, la edad, que tengan en el indicativo el número 4 (manejo de substrings)

```
# cat ordenes
{
FS = " "
OFS = "                "
}
($2 == "F") && (substr($4,1,3) == "(4)") {print $1 ,$3, $4}

# awk -f ordenes nombres
```

diana	9	(4)6313142	
patricia	35	(4)550909	

5. ADMINISTRACIÓN GENERAL DE MAQUINAS UNIX

Todo sistema UNIX debe tener al menos una persona encargada del mantenimiento y operación del sistema, que guíe a los usuarios para un buen manejo de los dispositivos involucrados en un ambiente multiusuario, que se haga responsable de las labores diarias de backups, de la verificación de espacio libre en el disco duro, borrado de archivos basura, del control de acceso al sistema como super-usuario etc. Esta persona es llamada "Administrador del sistema".

El manejo del sistema y la solución de problemas, requieren de un conocimiento considerable del trabajo interno de la máquina.
En otros documentos nos crentraremos en los casos particulares de los sistemas operativos Solaris, Iris y Linux. Aquí mencionaré conceptos comunes a todos los sistemas unix.

5.1. PROCEDIMIENTO GENERAL DE ARRANQUE

5.1.1. PROCESO DE BOOT DEL SISTEMA

Antes de que se tenga a disposición el Sistema Operacional uix, se desarrollan una serie de tareas que tienen que ver con el proceso de arranque de la máquina y del sistema . Estas si difieren mucho del tipo de hardware y sistema operativo.

Para poder comprender estos conceptos , debemos tener en cuenta ciertos conceptos .

5.1.1.1. DISCOS Y PARTICIONES

El disco del sistema contiene archivos y programas necesarios para bootear y cargar el sistema operativo . Este disco es divido en secciones llamadas particiones , las cuales detallaremos más adelante , pero que mencionaremos a continuación . Básicamente se necesitan 3 particiones (los números asignados , por defecto están entre “()”):

- root (0) : Contiene el sistema de archivos del / . En este están los archivos y directorios esenciales para cargar unix . También puede contener el subdirectorio /usr , donde reside el resto del sistema operativo , si es que este no está montado en una partición separada. Es conveniente tener esta última opción.
- swap (1): Destinada para paginación .
- usr (6) : es opcional , pues puede ser un subdirectorio de / , como se mencionó anteriormente .

En la sección de manejo de discos , ampliaré la información al respecto .

5.1.1.2. AMBIENTES DE LA MAQUINA

Las máquinas unix, tienen varios ambientes básicos de trabajo :

MODO MANTENIMIENTO.

En las máquinas RISC, cuando el sistema arranca , realiza una serie de diagnósticos, y presenta la opción de entrar a un modo mantenimiento, para realizar una serie de tareas que facilitan las labores de instalación , recuperación , diagnósticos y cargue del sistema . En Iris, con la tecla escape (Esc) , presionada instantes después del arranque , se podrá acceder este menú o pantalla gráfica , según sea el caso (servidor con consola texto , o estación con monitor gráfico) .En Solaris presionando ctrl.-stop.

En las máquinas intel con Linux, se dispone de un ambiente muy sencillo, que permite decidir algunas acciones a tomar, antes del proceso de cargue normal de la máquina.

UNIX.

Es el corazón del sistema operativo , frecuentemente referido como el kernel. Esta en el disco, generalmente en la partición (0) o del root. Al ser booteado, es cargado a memoria del sistema.

Unix controla el acceso a los dispositivos de hardware, crea programas que permiten el multiprocesamiento y el ambiente multiusuario. Estos programas

permiten limitar los recursos (cpu, memoria, disco) , para compartirlos productivamente entre muchos programas y usuarios .

Entre los programas que arranca , están :

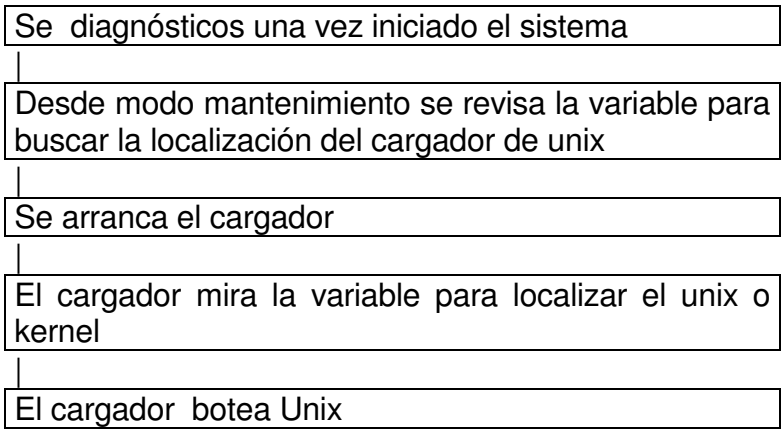
- sched. Llamado también scheduler, maneja el time slicing (timepo compartido) , decide quién y cuanto tiempo , tiene control sobre la cpu en un instante dado .
- vhand. Virtual memory handler, maneja el swaping.
- bdflush . Buffer to disk flush , buffer caché . Todos los I/O de disco son temporalemente almacenados en memoria antes de ir al disco, para reducir excesivo I/O . Este programa realiza este movimiento .
- init . Establece los niveles de unix y permite el cambio entre ellos . Crea y elimina una serie de procesos .

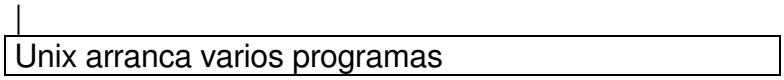
Este proceso de cargue de unix , se empieza desde el modo mantenimiento y termina cuando el sistema llega al nivel multiusuario o el especificado en el archivo respectivo.

En este nivel se tienen todos los servicios disponibles y se debe proceder a dar una cuenta (login) para trabajar .

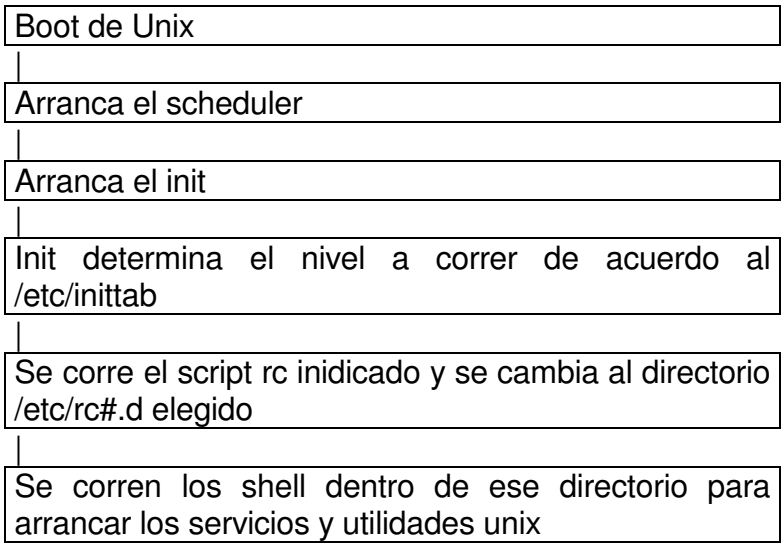
5.1.2. DIAGRAMA GENERAL DEL PROCESO DE BOOT

Para llegar al cargue de unix se realizan los siguientes pasos :





Una vez unix sea booteado , se ejecutan otras tareas :



5.1.3. ESTADOS O NIVELES UNIX

El Sistema Operacional Unix tiene varios estados:

0 El sistema se apagará.

1,**s** El sistema trabajará en modo monousuario.

2 Modo multiusuario.

3 Modo multiusuario y usuarios remotos en solaris

5 En linux es el modo multiusuario con ambiente gráfico

6 El sistema se apagará y reiniciará automáticamente.

Para pasar de uno a otro estado, siempre acudirá a la tabla /**etc**/**inittab**, en donde encontrará el procedimiento a seguir para llegar a dicho estado. Desde sistema operativo , podemos cambiar de estado con el comando init , o shutdown , como se verá más adelante .

5.1.4. PROCESOS INICIALIZADOS EN EL ARRANQUE DEL SISTEMA Y ARCHIVOS INVOLUCRADOS.

Después del test de hardware, el sistema realiza los siguientes pasos:

- **boot**: Ejecuta el programa boot, que se encarga de subir a memoria principal el archivo **unix**.

- **init**: El unix se encarga de ejecutar el programa **init**, que a su vez verifica en el archivo /etc/inittab, cual es el estado por defecto en el que debe quedar el sistema.

- /**etc**/**inittab**: En esta tabla de inicialización, además se encuentra el estado en el que quedarán las líneas directas del sistema.

- /**etc**/**rc2**: Una vez identifica el estado por defecto, ejecuta el programa que le corresponde, por ejemplo **rc2**, el cual a su vez busca el directorio que le corresponde a dicho estado.

- /**etc**/**rc2.d**: Por ejemplo para el estado multiusuario 2, le corresponde el directorio /etc/rc2.d en el cual se encuentran los siguientes procesos:

    - Actualización y montaje de sistemas de archivos.
    - Ejecuta el daemon **cron**.
    - Presentación de la actual configuración de memoria.

- Hacer disponible el **uucp**.

- Activar el scheduler para impresoras.

- Activar consola.

- Activar usuarios.

En este directorio se puede incluir los procesos que el administrador del sistema requiera. La estructura del nombre es {**KS**}{**nn**}**nombre**, donde:

K o **S** Kill para matar procesos o Start para arrancar.
nn Número de dos cifras, indica el orden de ejecución.
nombre Nombre con que se identificará el proceso.

Todos estos archivos generalmente invocan procesos cuyos fuentes se encuentran en el directorio /**etc**/**init.d**.

5.1.5. APAGADO DEL EQUIPO

Existen varias formas de bajar el sistema. A través del comando:

shutdown [**-y -g**seg **-i**estado]

donde:

y: Responde automáticamente **yes** a todas las preguntas.

g: Permite definir a los cuantos segundos se bajará el sistema. (defecto 60 seg) (período de gracia).

i: Permite identificar a que estado se llevará el sistema. (por defecto **s**).

La forma más rápida y sencilla es ejecutando el comando:

init 0

El cual ejecuta el proceso /etc/rc0, que ejecuta los shell encontrados en el directorio /etc/rc0.d, que empiezan por **K**, e invocan los respectivos archivos en /etc/init.d.

Algunos de los procesos en /etc/rc0.d son:

- Chequea que el usuario este en /

- Chequea que el usuario sea root

- Asigna el período de gracia 60seg

- Advierte a todos los usuarios que el sistema se esta bajando.

- Mata todos los procesos que se están ejecutando

- Desmonta los sistemas de archivos

- Invoca el estado 0.

En el proceso de apagado del equipo se pueden tener varias situaciones para la cuales se pueden elegir ciertos comandos :

- Se desea bajar el sistema completamente e ir al modo mantenimiento , para lo cual se pueden utilizar cualquiera de los siguientes comandos :

  init 0
  shutdown -y -g0

- Se desea pasar la máquina a modo monousuario :

  init 1

init s
shutdown –g0 -is

- Se desea rebootear la máquina (bajar y subir el sistema)

  init 6
  reboot

5.2. ALGUNOS PROCEDIMIENTOS MANUALES

5.2.1. ADICIONANDO USUARIOS

Para adicionar usuarios es necesario tener en cuenta los siguientes parámetros:

- Nombre del login con el que el usuario se identificará ante el sistema.
- Número de dicho login en el sistema (es el UID, no puede ser repetido y genealmente debe ser mayor de 100).
- Grupo al que pertenecerá dicho usuario (grupo primario, pues los secundarios se especifican en el archivo group).
- Directorio HOME del usuario (este hay que crearlo manualmente y asignarle los permisos necesarios).
- Shell que correrá el usuario (dependiendo de este se debe copiar un template del respectivo archivo profile requerido).

Los archivos que se modifican al crear un usuario son:

/etc/passwd

/etc/shadow
/etc/group

El archivo /**etc/passwd** tiene la siguiente estructura:

```
root:x:0:1:0000-Admin(0000):/:
daemon:x:1:1:0000-Admin(0000):/:
bin:x:2:2:0000-Admin(0000):/bin:
sys:x:3:3:0000-Admin(0000):/usr/src:
adm:x:4:4:0000-Admin(0000):/usr/adm:
uucp:x:5:5:0000-uucp(0000):/usr/lib/uucp:
nobody:x:14:14:for weak daemons:/tmp:
rje:x:18:18:0000-rje(0000):/usr/rje:
lp:x:71:2:0000-lp(0000):/usr/spool/lp:
man:x:99:1:0000-Admin(0000):/:
setup:x:0:0:general system administration:/usr/admin:/bin/rsh
sysadm:x:0:0:general system administration:/usr/admin:/bin/rsh
piadm:x:0:20:Prime INFORMATION PLUS Administrator:/usr/isys:/bin/sh
cata:x:106:1:cata:/usr/people/cata:
ttempo:x:107:1:ttempo:/usr/people/ttempo:
hector:x:108:1:hector gil triana:/usr/people/hector:/bin/sh

 1   2 3 4       5              6         7
```

En donde:

1. Nombre del login: Debe tener de 3 a 6 caracteres, el primero debe ser letra, los demás alfanuméricos en minúsculas.

2. Password: Este espacio es vacio si no tiene password. Si tiene aparece una **x**. El encriptado se encuentra en el directorio /**etc/shadow**.

3. Número del usuario: Un único número de 100 a 50.000, del 0-99 son del sistema.

4. Número del grupo: Un número de 100 a 50.000, del 0-99 son del sistema.

5. Comentarios: Identificación completa del usuario, nombre o alguna descripción.

6. Directorio HOME: PATH completo del directorio HOME de dicho usuario.

7. Shell: Este campo define el programa a ejecutar por defecto es el /bin/sh o el /bin/bash en linux.

5.2.2. MANEJO DE GRUPOS

Semejante al procedimiento con usuarios , se deben incluir la línea con los campos respectivos en el archivo /**etc**/**group** , el cual tiene la siguiente estructura:

```
root::0:root,hector
other::1:root
bin::2:root,bin,daemon,hector
sys::3:root,bin,sys,adm,hector
adm::4:root,adm,daemon,hector
uucp::5:root,adm,bin,uucp,hector
mail::6:root,hector
rje::8:rje,hector
nuucp::10:root,nuucp,uucp
daemon::12:root,daemon,hector
nobody::14:root,nobody,hector

 1   2 3       4
```

En donde:

1. Nombre del Grupo: Debe tener de 3 a 6 caracteres, el primero letra y los demás alfanuméricos en minúscula.

2. Password: No se usa

3. Número del Grupo: Un número único de 100 a 50.000, del 0 al 99 son del sistema.

4. Lista de usuarios: Lista de login de usuarios que pertenecen a dicho grupo.

5.2.3. PASSWORD ENCRIPTADOS

Para proteger las cuentas de los usuarios , ya que el archivo passwd tiene permisos de lectura para gran parte de usuarios, y podría ser un hueco en la seguridad de las claves , pues si allí se almacena el password encriptado , cualquiera podría leerlo y a través de alguna herramienta para desencriptar podría encontrar las claves de los usuarios, se implementa el archivo /**etc**/**shadow** que es accesado solamente por el sistema para crear y mantener claves de usuarios y está protegido con permisos (r--------) y su dueño es root . Así sólo puede ser leído y modificado por el usuario **root**. Su estructura es la siguiente:

```
root:VP0Mz3ykQw8GU:8456:: : :
daemon:*:8456:: : :
bin:*:8456:: : :
sys:*:8456:: : :
adm:*:8456:: : :
uucp:*:8456:: : :
lp:*:8456:: : :
man:*:8456:: : :
setup::8456:: : :
sysadm::8456:: : :
piadm:/GAcAiRtwxyvE:8456:: : :
pro::8456:: : :
cata::8456:: : :
ttempo::8456:: : :
hector:LuZNmi70V6ip2:8456:: : :
```

1 2 3

En donde:

1.Nombre del usuario: Login del usuario.

2.Password Encriptado: 13 caracteres que encriptan el password del usuario. Si esta presente *****, indica que el login no se puede usar. Si no hay nada, el login no tiene password.

3.Número: Número de días a partir de 4/4/70 hasta la última modificación del password.

Los siguientes campos son usados para la vigencia del password:

Mínimo: Número de días mínimo requerido para cambiar el password.

Máximo: Máximo número de días que el password es válido.

Algunos comandos para el manejo de estos archivo son:

pwck Chequea la sintaxis y la asociación lógica de cada usuario en el archivo /**etc**/**password.**

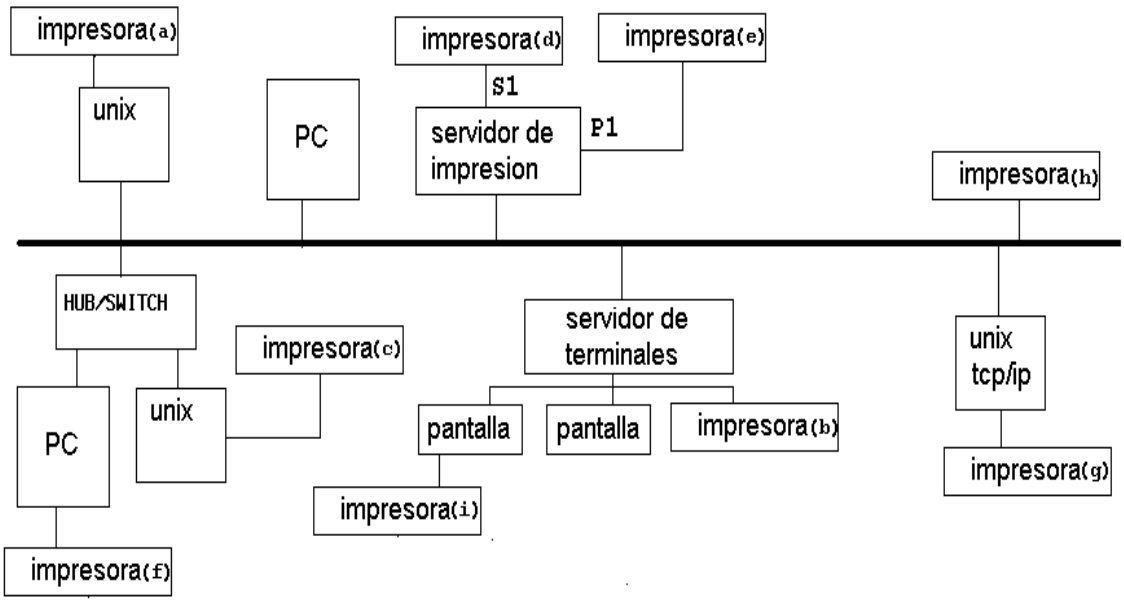
grpck Chequea el contenido de /**etc**/**group.**

pwconv Como inicialmente /etc/shadow no existe, este comando lo crea a partir de /**etc**/**passwd** si ya existe, simplemente lo actualiza.

newgrp [-][nombre_grupo]
Permite cambiar de grupo al usuario.Sin opciones retorna a su grupo primario. La opción "-" reprocesa el .profile. Para cambiar de grupo, el usuario debe pertenecer a la lista de logins (usuarios) de dicho grupo.

5.2.4 CONFIGURACION DE PERIFERICOS (IMPRESORAS)

En los sistemas de computo actuales , existe gran variedad de opciones para colocar una impresora y que esta sea accesada desde un equipo . En la siguiente gráfica trataré de esbozar las posibles formas de tener una impresora accequible desde un sistema unix.



Vemos en este diagrama muchas posibilidades de conexión , pero las podemos resumir en tres clases principales. Las que llamaré locales (a,b) , de red (c,d,e,f,g,h) y esclavas (i). Estas se pueden manejar con el spool normal de unix system v , administrador gráfico de impresión, spool de bsd, o secuencias de escape para impresoras esclavas . A continuación describiré cada una de estas opciones ; para lo cual llamaré a cada impresora por la letra que le corresponde.

5.2.4.1. SPOOL NORMAL DE UNIX SYSTEM V.

Los archivos enviados a impresión por este método utilizan el comando **lp** para imprimir y **cancel** para eliminar un trabajo de la cola de impresión. Ejemplo :

lp /etc/group Para enviar a la impresora default

lp -dlaser /etc/group Para enviar por una impresora determinada

Cuando se ejecuta esta órden , el sistema responde con el número de petición asignada a su trabajo :

request is laser-13

Para cancelar un trabajo se debe hacer referencia a este número , así :

cancel laser-13

Las tareas de operación y administración se reducen básicamente a crear la impresora , activarla , recibir trabajos en la cola , definir clases de impresoras si es el caso .

Si se tiene un monitor gráfico o un PC con un software para emularlo , es más sencillo invocar el print manager para ello , pero lo trataremos más adelante . Los archivos y directorios involucrados para el manejo del spool se encuentra bajo el directorio **/usr/spool/lp,** o **/var/spool/lp** .

Algunos comandos para el manejo de este spool son:

lpsched: Inicializa el spooler. Si se desea ejecutar manualmente se debe dar el siguiente comando con el pathname completo , o incluir el directorio /usr/lib dentro de la variable PATH:

/usr/lib/lpsched

Para que se ejecute automáticamente con la subida del sistema, existe un archivo de arranque en /etc/init.d . Sin este servicio activo , no se pueden invocar comandos para ver estado de impresoras , o enviar listados a impresoras .

lpshut Detiene el spool. Todas las impresoras quedan inmediatamente bloqueadas, inclusive las que estén imprimiendo en dicho momento.
Al ejecutar lpsched se activan todas las impresoras.

Es conveniente correr este comando (lpshut) antes de configurar una impresora con lpadmin:

/usr/lib/lpshut

lpmove: Permite cambiar archivos en cola de impresión, de una a otra impresora o trasladar toda la cola de impresión completa .

Formato:

lpmove archivo destino Mueve archivo a nuevo destino

lpmove destino.1 destino.2 Mueve todos los archivos de destino.1 a destino.2

accept: Permite que la impresora designada acepte archivos en cola de impresión o abre la cola de impresión (El pathname completo es /usr/lib/accept)

Formato:

accept impresora1

reject: Bloquea la impresora designada, no acepta archivos en la cola de impresión o la cierra (Está en /usr/lib/reject)

Formato:

reject impresora1

enable: Habilita la impresión de archivos en la cola (Está en /usr/bin/enable)

Formato:

enable impresora(s)

disable: Congela los archivos que están en cola o detiene su impresión.

Formato:

disable [**-c**] [**-r**[comentario]] impresora

donde:

-c: Cancela cualquier archivo que este en la cola de impresión.

-r: Asociado con un comentario, que explica porque se congelo esa impresora.

lpadmin: Permite crear, borrar impresoras. Además permite cambiar la impresora default. El comando se encuentra en /**usr/lib**.

formato:

lpadmin -pdestino opciones

lpadmin -xdestino

lpadmin -ddestino

En donde:

d: define la nueva impresora default.

x: Borra la impresora especificada.

p: Configura una impresora, con las opciones especificadas a continuación. Los argumentos se dan a continuación de las opciones sin dejar espacio en blanco(se debe haber deshabilitado antes el scheduler):

-v: Especifica el pathanme del dispositivo en el cual se ubica la impresora. Ejemplo : /dev/ttyd2

-m: Especifica el modelo de la impresora que se utilizará como interface. Ejemplo: dumb

-e: Se usa para especificar una interface existente para la nueva impresora.

-c: Asigna una clase a dicha impresora.

-r: Borra una impresora de una clase.

Ejemplo : Crear la impresora prueba por el puerto paralelo (dispositivo /dev/plp) , usando el interface dumb (apropiado para impresoras de texto) :

/**usr**/**lib**/**lpadmin** -**p**prueba -**m**dump **-v**/dev/plp

/**usr**/**lib**/**accept** prueba

/**usr**/**bin**/**enable** prueba

Para monitoreo del estado del sistema de spool y de las impresoras se utilizan los comandos :

lpstat -r	Muestra si el spool está corriendo o no .
lpstat -d	Muestra la impresora que está definida como default
lpstat -o	Muestra la cola de impresión global . Se puede opcionalmente colocar después de la opción **o** el nombre de la impresora de la cual deseo ver la cola de impresión .
lpstat -p	Presenta las impresoras definidas en el sistema y su estado . Es decir si están ociosas, imprimiendo , deshabilitadas , etc .
lpstat -v	Presenta las impresoras definidas en el sistema y los dispositivos a los cuales ellas están conectadas .
lpstat -t	Presenta toda la información anterior .

Ejemplos :

Ver toda la información concerniente al sistema de spool local :

```
#lpstat -t
scheduler is running
system default destination: mediop
device for mediop: /dev/ttyq24
device for impre2: /dev/null
```

```
device for impre3: /dev/null
device for impre1: /dev/null
device for anchop: /dev/ttyq23
impre1 accepting requests since May 22 04:26
impre2 accepting requests since May 22 04:27
impre3 accepting requests since May 22 04:27
anchop accepting requests since Sep 25 16:41
mediop accepting requests since Sep 25 16:52
printer impre1 is idle.  enabled since May 22 04:27
printer impre2 disabled since May 22 04:27 -
        reason unknown
printer impre3 disabled since May 22 04:27 -
        reason unknown
printer anchop disabled since Sep 25 17:24 -
        reason unknown
printer mediop is idle.  enabled since Sep 25 16:52
impre3-2 root   10094   May 22 04:28
impre3-3 curso3   10094   May 22 04:28
impre3-4 root    1123   May 22 04:28
impre3-5 root    4670   May 22 04:29
impre2-6 curso3    2991   May 22 04:29
impre3-7 curso1    1123   May 22 04:29
impre2-8 curso3   10094   May 22 04:30
impre3-9 curso5   11470   May 22 04:30
impre3-10 curso5   10094   May 22 04:30
impre2-11 root    4670   May 22 04:31
impre2-12 root    1523   May 22 04:31
impre2-13 curso3   10094   May 22 04:34

#lpstat -p

printer impre1 is idle.  enabled since May 22 04:27
printer impre2 disabled since May 22 04:27 -
        reason unknown
printer impre3 disabled since May 22 04:27 -
        reason unknown
printer anchop disabled since Sep 25 17:24 -
        reason unknown
printer mediop is idle.  enabled since Sep 25 16:52
```

#lpstat -oimpre3

impre3-2 root 10094 May 22 04:28
impre3-3 curso3 10094 May 22 04:28
impre3-4 root 1123 May 22 04:28
impre3-5 root 4670 May 22 04:29
impre3-7 curso1 1123 May 22 04:29
impre3-9 curso5 11470 May 22 04:30
impre3-10 curso5 10094 May 22 04:30

#lpstat -o

impre3-2 root 10094 May 22 04:28
impre3-3 curso3 10094 May 22 04:28
impre3-4 root 1123 May 22 04:28
impre3-5 root 4670 May 22 04:29
impre2-6 curso3 2991 May 22 04:29
impre3-7 curso1 1123 May 22 04:29
impre2-8 curso3 10094 May 22 04:30
impre3-9 curso5 11470 May 22 04:30
impre3-10 curso5 10094 May 22 04:30
impre2-11 root 4670 May 22 04:31
impre2-12 root 1523 May 22 04:31
impre2-13 curso3 10094 May 22 04:34

#lpstat -r

scheduler is running

#lpstat -v

device for mediop: /dev/ttyq24
device for impre2: /dev/null
device for impre3: /dev/null
device for impre1: /dev/null
device for anchop: /dev/ttyq23

#lpstat -d

system default destination: mediop

```
#
```

5.2.4.2. SPOOL DE BSD.

Es utilizado para el acceso a impresoras de red . Los comandos para envío y operación son diferentes a los otros y los detallaremos a continuación .

El envio de listados a impresión se realiza mediante el comando **lpr** , y si se desea especificar una impresora en particular , se utiliza la opción **-P** . Para eliminar un trabajo de la cola de impresión se utiliza el comando **lprm** y el número de petición del trabajo . También se utiliza la opción **-P** para especificar la impresora deseada , si no es la por defecto . El envió , cancelación y consulta de la cola del spool se ilustraran en el siguiente ejemplo :

Se enviará a imprimir en la impresora llamada hplce, el archivo /etc/passwd y después se consultará la cola del spool (con el comando lpq) , para proceder a removerlo (con lprm). La cola de impresión para esa impresora está detenida , para efectos de poder mostrarla sin evacuar listados .

```
# lpr -Phplce /etc/passwd

# lpq -Phplce

Warning: hplce queue is turned off
Warning: no daemon present
Rank   Owner      Job  Files                          Total Size
1st    root       0    /etc/group                     411 bytes
2nd    root       1    /etc/passwd                    871488 bytes

# lprm -Phplce 0

dfA000www.unab.edu.co dequeued
cfA000www.unab.edu.co dequeued

# lpq -Phplce

Warning: hplce queue is turned off
Warning: no daemon present
Rank   Owner      Job  Files                          Total Size
```

```
1st    root      1    /etc/passwd                    871488 bytes

# lprm -Phplce 1

dfA001www.unab.edu.co dequeued
cfA001www.unab.edu.co dequeued

# lpq -Phplce

Warning: hplce queue is turned off
no entries
```

Los comandos y archivos involucrados en la creación y operación de este tipo de impresoras se describirán a continuación :

- Para la creación manual de estas impresoras , se debe verificar que este sistema de impresión esta soportado en el sistema. En Iris se verifica :

```
# versions |grep bsd

I  print.sw.bsdlpr      08/27/97  Berkeley 'lpr' Printer Spooler
```

En Solaris 9 esta soportado por el sistema (los dos tipos de ambientes de impresión)

- Editar el archivo /etc/printcap , que contiene las definiciones de estas impresoras e incluir las líneas respectivas para su creación . El significado de los parámetros a incluir en este archivo son :

```
      nombre-impresora| comentarios :\
 :rm=nombre-maquina-remota-que-contiene-impreora:\
 :rp=nombre-del-puerto-remoto-o-cola-de-impresion-remota:\
 :sd=/var/spool/lpd/nombre-directorio:
```

El nombre de la impresora , es el nombre local , con e cual se desea referenciar la impresora remota. No necesariamente debe ser el nombre real o remoto de la impresora.

El nombre de la máquina remota , a donde esta conectada la impresora , debe estar incluido con su respectiva dirección IP en el archivo /etc/hosts . En caso

contrario se puede incluir directamente la dirección IP de la máquina en este campo.

En el caso de los servidores de impresión, donde sus puertos (seriales y paralelos) se identifican por un nombre de servicio , este , se debe colocar a continuación del parámetro **rp=** . Para impresoras conectadas a servidores , o PCs que pueden compartir sus impresoras con la red (bajo tcp/ip) , en ese parámetro se especifica el nombre de la cola de impresión o impresora remota.

Con el parámetro **sd=** se especifica el nombre del directorio local , donde se desea se almacene la cola de impresión . En ese directorio de deben tener ciertos permisos .

un ejemplo de archivo printcap es el siguiente :

```
# cd /etc

# cat printcap

#
# Copyright (c) 1983 Regents of the University of California.
# All rights reserved.
#
# Redistribution and use in source and binary forms are permitted
# provided that this notice is preserved and that due credit is given
# to the University of California at Berkeley. The name of the University
# may not be used to endorse or promote products derived from this
# software without specific prior written permission. This software
# is provided ``as is'' without express or implied warranty.
#
#       @(#)etc.printcap        5.2 (Berkeley) 5/5/88
#
# DecWriter over a tty line.
#lp|ap|arpa|ucbarpa|LA-180 DecWriter III:\
#       :br#1200:fs#06320:tr=\f:of=/usr/lib/lpf:lf=/var/adm/lpd-errs:
# typical remote printer entry
#ucbvax|vax|vx|ucbvax line printer:\
#       :lp=:rm=ucbvax:sd=/var/spool/vaxlpd:lf=/var/adm/lpd-errs:
```

```
hplce:\
    :rm=sunab18:\
    :sd=/var/spool/lpd/hplce:

p5000:\
    :rm=M_030d76:\
    :rp=d1prn:\
    :sd=/var/spool/lpd/p5000:
```

- Crear el directorio para la cola del spool y dar los permisos necesarios . Ejemplo :

```
#cd /var/spool/lpd

#mkdir p5000

# chmod 777 p5000
```

- Proceder con los comandos necesarios para operación de este sistema de spool . Son muy semejantes a los del sistema de spool anterior y se pueden invocar de dos formas : Desde la línea de comandos del **lpc** con opciones y argumentos , o dentro de un nivel de subcomandos del comando **lpc** .

Los comandos más utilizados para la manipulación de estas impresoras son :

lpc : Permite dar los subcomandos para operación del spool .

lpq : Permite revisar el estado de la cola de impresión de una impresora.

Desde el diálogo del **lpc** , se pueden dar otros subcomandos para habilitar, detener , reiniciar ,etc las impresoras (especificadas como argumento) , tales como :

abort : Abortar la impresión de un trabajo
enable : Habilitar la impresión de trabajos
disable : Deshabilita la impresión de trabajos
help o ? : Presenta la ayuda

restart : Reinicia el trabajo de una impresora y su daemon
status : Presenta el status de una o todas las impresoras
exit : Salir
quit : Salir
start : Arranca el servicio de una impresora y su daemon
stop : Detiene el servicio de una impresora y su daemon

Ejemplos :

```
# lpc status

hplce:
        queuing is enabled
        printing is enabled
        1 entry in spool area
        no daemon present
p5000:
        queuing is enabled
        printing is enabled
        2 entries in spool area
        no daemon present

www 6# lpq -Phplce

Warning: no daemon present
Rank   Owner      Job  Files                                 Total Size
1st    root       0    /etc/group                            411 bytes

www 7# lpc

lpc> ?
Commands may be abbreviated.  Commands are:

abort   enable  disable help    restart status  topq    ?
clean   exit    down    quit    start   stop    up
lpc> stop hplce
hplce:
        printing disabled
```

```
lpc> enable hplce
hplce:
        queuing enabled

lpc> disable hplce
hplce:
        queuing disabled

lpc> start hplce
hplce:
        printing enabled
        daemon started

lpc> quit
```

5.2.4.3. CONFIGURACION DE IMPRESORAS ESCLAVAS

El manejo de las impresoras esclavas es un poco diferente y depende más de secuencias de control de cada tipo de terminal . Estas secuencias están especificadas en los manuales de programación respectivo , de cada pantalla.

Al dar cierta secuencia ,la pantalla queda en lo que llaman modo transparente y cualquier comando y su respectiva respuesta es direccionada al puerto auxiliar de la pantalla . Dependiendo de la pantalla este puerto auxiliar puede ser serial , o serial y paralelo como en las wyse150 . En ese puerto auxiliar se conecta la impresora esclava .

En últimas lo que se hace una vez este activado el modo transparente , es hacer un "cat" del archivo a imprimir y este es direccionado al puerto auxiliar y así a la impresora . Una vez terminado el listado se debe dar la secuencia de escape para anular el modo transparente, pues sino la pantalla quedaría bloqueada y todos los comandos y respuestas de ahí en adelante saldrían por la impresora . Cabe resaltar que estas configuraciones son para imprimir listados de pocas hojas (1, 2 o 3) pues mientras imprime la pantalla queda bloqueada .

Se debe crear un archivo en la máquina unix que contenga las siguientes lineas :(al frente las describo y aplica para pantallas VT100) :

```
stty echo          /* deshabilitar el echo de caracteres por la pantalla
echo "\033[5i"     /* secuencia de escape para las VT100 modo transparente
cat $1             /*  listado del archivo que se coloque como argumento
echo "\033[4i"     /*  secuencia para deshabilitar modo transparente
stty -echo         /*  Volver a habilitar echo de caracteres por pantalla
```

Si este archivo se llama esclava , se debe grabar en /sbin , pues es accesible por todos los usuarios y darle los permisos 755 . Para poder imprimir en cualquier impresora esclava , se digitaria :

esclava nombre-archivo-a-imprimir

Como puede observar la secuencia de escape varía , de acuerdo con la pantalla que este utilizando . Es diferente para wy60, ansi , etc...

5.2.5. CONFIGURACIÓN DE LINEAS PARA TERMINALES O MODEM .

Aunque ya no es muy común utilizar los puertos seriales que tienen algunos servidores, mencionaremos esto.
En el archivo /**etc/inittab** escoja la línea a activar, cambie su estado **off** a **respawn**. A continuación reinicialice dicho archivo ejecutando :

telinit q

Comandos para el manejo de líneas:

getty Este proceso es creado por init y establece tipo de terminal, modo, velocidad y disciplina de la línea. Imprime el mensaje de login especificado en el archivo **gettydefs**.

Es utilizado en conjunto con /etc/inittab.

formato básico:

getty [**-t** seg] línea [speed[tipo]]

Sin opciones, toma como definición por defecto la primera línea en /etc/gettydefs.

-t Define el tiempo en segundos, en el cual se desactivará la línea si no se percibe ningún movimiento de información.

línea Línea del directorio /dev, a la que se le definirán los parámetros.

speed Determina la velocidad de dicha línea, por default es 300 baud.

tipo Especifica el tipo de terminal conectada a esa línea.

Otra forma del comando es:

/etc/getty -c /etc/gettydefs

Hace seguimiento al archivo gettydefs y reporta posibles errores.

5. 3. COMANDOS PARA BACKUPS

En todos los sistemas de computo , es indispensable tener copia de la información , en algún medio , para que sea recuperada en el momento que así se requiera . Dentro de las políticas de backups o de seguridad de la información , se puede pensar en :

- Tener una copia total de la información en un medio magnético (backup completo, podría ser quincenal o mensual) y luego sacar backup de la información que ha sido modificada , desde la copia anterior (backup incremetal, pueden ser diarios). Esto con el fin de disminuir el tiempo de backup , ya que los incrementales son mucho más cortos . Aunque , en el proceso de recuperación , puede ser mucho más demorado .
- Sacar siempre backups completos . Estos han disminuido en tiempo gracias a la evolución de los medios magnéticos , y dependen mucho de la cantidad de información a almacenar.

- Pensar en sistemas de disco organizados en lo que se conoce como Mirror (discos espejo) , en los cuales siempre existe la información duplicada en otro disco , que remplazaría a la del disco original , en el caso que se requiera.
- Existen otras formas de organización de los discos , que permiten tener un sistema confiable , pero tanto en la anterior , como en estas , es recomendable pensar en los backup en algún medio magnetico/óptico .

Recordemos que hay diferentes tipos de medio magnético , para la realización de los backups . Existen tape Exabyte o de 8 mm , unidades de tape DAT o de 4mm, arreglos de unidades con capacidades superiores , etc. Lo que difiere en los sistemas unix es la forma como son nombradas las unidades.

Generalmente las unidades de almacenamiento , son encontradas en el directorio /dev/rmt y su nombre depende del controlador y la posición en la cual se conectan .

Supongamos que hay una unidad de backup DAT (4mm) , conectada al controlador scsi 1 , en la posición 6 . De esa posición depende el nombre que a ella se le asigna . Si el sistema tiene la nomenclatura :

tpsN**d**MPPP

donde :
N : Número del controlador a donde esta conectada la unidad .
M : Posición en el controlador en donde se conectó la unidad.
PPP : Puede tener los siguientes valores :
nr : El dispositivo no rebobina después de una operación
v : El dispositivo puede manejar registros de longitud variable.
C : Grabar en modo comprimido , duplicando su capacidad.

Existen más valores a tomar , pero no se detallarán . Con esta nomenclatura , el nombre de los dispositivos que se crean para un determinado medio magnético , pueden ser :

```
#pwd
/dev
#cd rmt
#ls
tps1d6      tps1d6nrnsv  tps1d6nrv   tps1d6s     tps1d6v
tps1d6nr    tps1d6nrs    tps1d6ns    tps1d6stat
```

```
tps1d6nrns   tps1d6nrsv   tps1d6nsv    tps1d6sv
```

Si deseáramos ver el estado de la unidad de cinta , se debe dar el comando :

```
mt -t /dev/rmt/tps1d6 status
      Controller: SCSI
      Device: SONY: SDT-9000       12.2
      Status: 0x200
      Drive type: DAT
      Media : Not READY
```

Para la realización de los backups , tenemos en varios comandos , los cuales discutiremos a continuación :

5.3.1 COMANDO TAR

Es usado para almacenar y restaurar archivos y directorios. Es más poderoso y útil cuando se hace backup de archivos simples o directorios, que cuando se hace de sistemas de archivos completos. El formato del comando es:

tar -opciones archivo1 archivo2 archivo3 ...

donde :

opciones : **c**: Escribe en la cinta.

x: Lee de la cinta.

v: Modo verbose.

t: Lista el contenido de la cinta.

u: Escribe solo si los nombres dados no existen en la cinta o si han sido modificados desde la última vez que fueron escritos.

f: Especifica que el siguiente argumento es el nombre del dispositivo a usar.

Ejemplos:

Para sacar backup del subdirectorio pruebas, dentro de /usr/people/hector:

```
#tar -cv /usr/people/hector/pruebas

a /usr/people/hector/pruebas/hector 17 blocks
a /usr/people/hector/pruebas/hector1 61 blocks
a /usr/people/hector/pruebas/hector2 9 blocks
a /usr/people/hector/pruebas/hector3 7 blocks
a /usr/people/hector/pruebas/hector4 17 blocks
a /usr/people/hector/pruebas/hector5 10 blocks
```

Para leer el archivo hector1 del backup hecho anteriormente:

```
# tar -xv /usr/people/hector/pruebas/hector1

x /usr/people/hector/pruebas/hector1, 30763 bytes, 61 tape blocks
```

5.3.2 COMANDO CPIO

Permite almacenar o restaurar de cinta archivos o directorios. Es más útil y poderoso para trabajar con archivos o directorios simples que con sistemas de archivos completos.

El formato del comando para almacenar es:

cpio -opciones **<**lista-de-nombres **>**dispositivo o

se puede dar la opción **-O** para especificar el dispositivo en el cual se realizará la copia (no se colocaría el signo >)

El formato para restaurar es:

cpio -opciones **<**dispositivo o

se puede dar la opción **-I** para especificar el dispositivo del cual se restaurará la información (no se colocaría el signo <)

Este comando comúnmente se usa en conjunción con el comando **find**, ya que este le proporciona la lista de nombres que el **cpio** va a almacenar.

donde :

opciones:

o: Escribir. la salida estándar debe ser direccionada al dispositivo a usar.

c: Escribe la información del header en formato ASCII.

d: Crea directorios si se requiere.

v: Modo verbose.

B: La salida es escrita en bloques de 5.120 bytes por registro. Esta opción aumenta la velocidad de la tarea. Si es usada para escribir, se debe usar también para leer.

i: Leer. Se direcciona el dispositivo del cual leer.

t: Lista sin restaurar.

Ejemplos:

Se realiza un backup del contenido del subdirectorio pruebas (Los archivos y directorios quedan con el pathname completo), en el dispositivo /dev/rmt/tps1d6 (DAT).

```
# find /usr/people/hector/pruebas/* -print | cpio-ocvdB >/dev/rmt/tps1d6
```

```
/usr/people/hector/pruebas/hector
/usr/people/hector/pruebas/hector1
/usr/people/hector/pruebas/hector2
/usr/people/hector/pruebas/hector3
/usr/people/hector/pruebas/hector4
/usr/people/hector/pruebas/hector5
120 blocks
```

Se podría haber usado el comando :

```
#find /usr/people/hector/pruebas/* -print | cpio-ocvdBO /dev/rmt/tps1d6
```

Es importante tener en cuenta si la información a almacenar esta en forma relativa o absoluta . En el caso anterior estaba en forma absoluta y de igual forma se tendría que recuperar , es decir se debe recuperar en la misma ubicación de filesystem y directorio .

Se puede realizar el mismo backup , pero los pathname son pasados en forma relativa y así al recuperarla , se puede hacer en otra ubicación .

```
# cd /usr/people/hector/pruebas
# find . -print |cpio -ovdcB >/dev/rmt/tps1d6
.
hector
hector1
hector2
hector3
hector4
hector5
120 blocks
```

Para sacar un índice del backup realizado anteriormente:

```
# cpio -itvd </dev/rmt/tps1d6

40755  root       0  Feb 25 15:38:04 1993  .
100644 root    8702  Feb 25 15:38:03 1993  hector
100644 root   30763  Feb 25 15:38:03 1993  hector1
100644 root    4355  Feb 25 15:38:04 1993  hector2
```

```
100644 root    3106  Feb 25 15:38:04 1993  hector3
100644 root    8265  Feb 25 15:38:04 1993  hector4
100644 root    4724  Feb 25 15:38:04 1993  hector5
119 blocks
```

5.3.3. COMANDOS XFSDUMP/UFSDUMP Y XFSRESTORE/UFSRESTORE

El nombre varia un poco del sistema utilizado. En el caso de particiones completas , existen herramientas como xfsdump y xfsrestore, que permiten realizar y recuperar backup de particiones individuales , y permiten una mayor interacción con el contenido de la cinta.

En versiones anteriores , donde existían sistemas de archivos EFS , este comando era conocido como **dump** . Yo lo llamaré por este nombre y ustedes lo acomodan al de la versión de su sistema.

La sintaxis del comando antuguo era :

dump opciones dispositivo sistema-de-archivos

Donde las opciones pueden ser :

: Número que representa el nivel del backup (0-9). Cero (0) es total .
d : Especifica la densidad de la unidad (expresada en bytes per inch BPI)
s : Especifica el tamaño de la cinta en pies .
f : Especifica la unidad donde se realizará el backup . Se debe especificar el dispositivo a continuación .

Para algunas de las versiones nuevas , el comando es dump cambia su sintaxis :

dump opciones sistema-de-archivos

Donde las opciones pueden ser :

-f : Especifica la unidad donde se realizará el backup . Se debe especificar el dispositivo a continuación .

-l : Especifica el nivel del backup , con un número de 0 a 9 .
-p : Especifica cada cuantos segundos se despliega información de la operación .
-v : Especifica si se muestran mensajes o no , para ver detalles en la operación.

Ejemplos : Para realizar un backup completo solamente de la partición del root (/) , con un nivel de detalle alto , mostrando mensajes por pantalla . Se pide inicialmente el nombre que se la dará a la cinta (label) y luego chequea si la cinta contiene información , en este caso que ya contenía , pide confirmación para sobrescribirla :

```
#dump -f /dev/rmt/tps1d7 -l 0 -p 30 -v trace /
xfsdump: version 2.0 - type ^C for status and control

 ============================dump label dialog ==============================

please enter label for this dump session (timeout in 300 sec)
 -> root
session label entered: "root"

 ---------------------------- end dialog -------------------------------

xfsdump: level 0 dump of pelicano.uis.edu.co:/
xfsdump: dump date: Tue Nov  3 14:56:53 1998
xfsdump: session id: 1731c081-07a3-1022-87d2-080069056a64
xfsdump: session label: "root"
xfsdump: ino map phase 1: skipping (no subtrees specified)
xfsdump: ino map phase 2: constructing initial dump list
xfsdump: ino map phase 3: skipping (no pruning necessary)
xfsdump: ino map phase 4: skipping (size estimated in phase 2)
xfsdump: ino map phase 5: skipping (only one dump stream)
xfsdump: ino map construction complete
xfsdump: estimated dump size: 38885440 bytes
xfsdump: /var/xfsdump/inventory created
xfsdump: preparing drive
xfsdump: fixed block size noncompressing tape drive at /dev/rmt/tps1d7
xfsdump: status at 14:57:22: 0/849 files dumped, 0.0% complete, 29 seconds elapsed
xfsdump: cannot determine tape block size after two tries
xfsdump: assuming is media is corrupt or contains non-xfsdump data
```

```
xfsdump: WARNING: media contains non-xfsdump data or a corrupt xfsdump media file header at beginning of media

 ============================media overwrite dialog ============================

overwrite non-xfsdump data on media in drive 0?
1: don't overwrite (timeout in 3600 sec)
2: overwrite (default)
 -> 2
media will be overwritten

 ------------------------------ end dialog --------------------------------

xfsdump: status at 14:57:52: 0/849 files dumped, 0.0% complete, 59 seconds elapsed

 ============================media label dialog ============================

please enter label for media in drive 0 (timeout in 300 sec)
 -> root
media label entered: "root"

 ------------------------------ end dialog --------------------------------

xfsdump: creating dump session media file 0 (media 0, file 0)
xfsdump: dumping ino map
xfsdump: dumping directories
xfsdump: dumping directory ino 128
xfsdump: dumping directory ino 131
xfsdump: dumping directory ino 135
xfsdump: dumping directory ino 136
xfsdump: dumping directory ino 137

.....        SE SUPRIMEN LINEAS , POR EFECTOS DE LONGITUD
.....
xfsdump: dumping special file ino 262334 mode 0xa1ed
xfsdump: dumping special file ino 262335 mode 0xa1ed
xfsdump: dumping special file ino 262368 mode 0xa1ed
```

```
xfsdump: dumping special file ino 262369 mode 0xa1ed
xfsdump: dumping regular file ino 262372 offset 0 to offset 1490 (size 1490)
xfsdump: dumping regular file ino 262373 offset 0 to offset 2482 (size 2482)
xfsdump: dumping regular file ino 262374 offset 0 to offset 680 (size 680)
xfsdump: dumping regular file ino 262375 offset 0 to offset 2038 (size 2038)

..... SE SUPRIMEN LINEAS , POR EFECTOS DE LONGITUD
.....
xfsdump: status at 14:59:52: 848/849 files dumped, 69.1% complete, 179 seconds elapsed
xfsdump: ending media file
xfsdump: media file size 39845888 bytes
xfsdump: dumping session inventory
xfsdump: beginning inventory media file
xfsdump: media file 1 (media 0, file 1)
xfsdump: ending inventory media file
xfsdump: status at 15:00:22: 849/849 files dumped, 96.3% complete, 209 seconds elapsed
xfsdump: dumping inventory
xfsdump: inventory media file size 4194304 bytes
xfsdump: writing stream terminator
xfsdump: beginning media stream terminator
xfsdump: media file 2 (media 0, file 2)
xfsdump: ending media stream terminator
xfsdump: media stream terminator size 2097152 bytes
xfsdump: ending stream: 217 seconds elapsed
xfsdump: I/O metrics: 3 by 2MB ring; 25/33 (76%) records streamed; 362235B/s
xfsdump: dump complete: 217 seconds elapsed
```

Para recuperar la información de la cinta se puede hacer de forma normal o interactiva . Para evitar discrepancias entre versiones de unix, yo lo llamaré restore. La sintaxis del comando restore es :

restore opciones directorio-destino

donde las opciones son :

-e : No sobrescriba archivos existentes en el disco
-f : Especifica el dispositivo del cual se va a extraer la información.

-i : Entra la modo de extracción interactivo . Tiene subcomandos .
-r : Extrae información
-t : Solamente lista contenido de la cinta . Si se escoje esta opción , nmo hay necesidad de especificar directorio destino .

Ejemplo : Se listará el contenido de la cinta grabada en el ejemplo anterior. Los subcomandos listados allí , aplican también si se va a extraer información de forma interactiva .

```
#xfsrestore -i -t -v silent -f /dev/rmt/tps1d7

 ========================dump selection dialog ============================

the following dump has been found on drive 0

hostname: pelicano.uis.edu.co
mount point: /
volume: /dev/rroot
session time: Tue Nov  3 14:56:53 1998
level: 0
session label: "root"
media label: "root"
file system id: 00000000-001d-9b40-a800-000000479800
session id: 1731c081-07a3-1022-87d2-080069056a64
media id: 1731c083-07a3-1022-87d2-080069056a64

examine this dump?
1: skip
2: restore (default)
 -> 2
this dump selected for restoral

 ------------------------------ end dialog --------------------------------

 ======================= subtree selection dialog ==========================

the following commands are available:
```

```
       pwd
       ls [ <path> ]
       cd [ <path> ]
       add [ <path> ]
       delete [ <path> ]
       extract
       quit
       help

 -> pwd
cwd is fs root

 -> ls
          155 .profile
         3957 Mensaje
       655508 INFORMIXTMP/
          947 backup.total.ag25
          969 .sh_history
          154 .login
       655499 lib64/
       655490 lib32/
       657696 .desktop-dsi15/
         3963 .expertInsight
          ..... SE SUPRIMEN LINEAS POR EFECTOS DE LONGITUD
                .....
         3947 unix
       657728 swap/
       524422 sbin/
          131 hw/

 -> cd swap

 -> ls
       657729 swap1

 -> cd ..

 -> pwd
cwd is fs root
```

```
-> cd etc

-> ls
        262410 rfind.aliases
        262310 prioinfo
        409345 config/
        262403 project
        262328 uncompvm
        262431 profile
        262428 gettydefs
        262304 nvram
                ..... SE SUPRIMEN LINEAS POR EFECTOS DE LONGITUD
                .....

 -> quit

---------------------------- end dialog --------------------------------
```

Los subcomandos a nivel del diálogo de la forma interactiva son :

pwd Indica el nombre del directorio donde se encuentra ubicado .
ls Lista el contenido de un directorio almacenado en la cinta .
cd Cambia de directorio en la cinta .
add Adiciona un directorio o archivo a la lista de elementos a extraer .
delete Elimina un directorio o archivo de la lista de elementos a extraer.
extract Empieza la recuperación de los elementos seleccionados .
quit Salir .
help Mostrar la ayuda

Para efectuar la recuperación de la información se podría haber invocado el comando sin la opción **-t** y dentro del diálogo seleccionar la información a recuperar , para luego dar **extract** .

También se podría recuperar en forma normal , con la opción -r , sin entrar al diálogo del modo interactivo .

5.3.4.MANEJO DE BACKUPS INCREMENTALES

Estos pueden ocupar menos espacio y tiempo que los totales . Un esquema de realización de backups incrementales para un sistema de archivos , puede ser :

- El primer día , sacar un baclup total del sistema de archivos completo (podría ser mensual).
- Del segundo al séptimo día , sacar backup de los archivos que han sido modificados el día anterior o backups diarios .
- El octavo día , sacar backup de los archivos modificados en la semana anterior o bacups semanal.
- Repetir los dos pasos anteriores por tres o cuatro semanas .
- Al mes repetir el primer paso .

En cada sistema particular pueden encontrar herramientas que facilitan estas labores, pero aquí mencionaremos la forma general de hacerlo.

BACKUP INCREMENTALES CON TAR Y CPIO

Aunque estos comandos en sí , no tienen mecanismos para backups incrementales, se pueden acompañar con otros para la realización de los mismos . Presentaré la forma de hacerlo , con el mismo esquema del caso anterior (la primera linea es la implementación con tar y la segunda linea es con cpio) :

- Cambiarse al sistema de archivos al que se le realizara el backup .

  ```
  #cd /usr
  ```

- Crear un backup completo de ese sistema de archivos.

  ```
  tar cv

  cpio -oLp .
  ```

- Cada día , sacar los backups diarios .

  ```
  find /usr -mtime 1 -print | tar cvf -
  ```

find /usr -mtime 1 -print | cpio -pdL

- Cada semana , los backups semanales .

  find /usr -mtime 7 -type f -print | tar cvf -

  find /usr -mtime 7 -type f -print | cpio -pdL

- Cada mes repetir el primer y segundo paso .

5.3.5.BACKUP AUTOMATICOS

Se puede utilizar la utilidad **cron** , para programar la ejecución de los mismos periodicamente . La cinta debe estar montada en la unidad y es aconsejable que todos los datos se puedan acomodar en la cinta para no requerir el cambio de la misma y así no tener intervención manual .

La siguiente línea debe ser incluida en al archivo : /**var**/**spool**/**cron**/**crontabs**/**root** o incluirla al involcar el comando **crontab -e**:

```
0 3  * *  *  tar -cvf /dev/rmt/tps1d6  /usr
```

En este caso se programa un backup completo de /usr , todas las mañanas a las 3 :00.

5.3.6. COMANDO MT

Es usado para dar órdenes y tener más interacción con la unidad de grabación. Permite la manipulación de la cinta magnetica en tareas como rebobinar , avanzar hacia adelante, atrás , definir tamaño de bloque, etc. Es utilizado junto con los comandos tar , cpio , bru para el manejo de múltiples cintas lógicas en una física .

La sintaxis general es :

```
mt  [ -t dispositivo ] comando  [contador]
```

Donde el dispositivo representa el archivo que identifica la unidad de grabación , tal como /dev/rmt/tps1d7 . En las situaciones donde se desee desplazarse sobre las cintas lógicas , es decir avanzar hacia adelante varias posiciones , o regresarse , se debe utilizar el dispositivo que no rebobina , para evitar que una vez efectuado el posicionamiento , la unidad rebobine la cinta y realmente no la deje en la posición deseada . El comando representa las órdenes que le daremos a la unidad , tal como rebobinar , desplararse hacia adelante , etc . El contador representa la cantidad de posiciones que se debe desplazar la unidad , en los comandos que así lo requieran .

Dentro de los comandos más usados se encuentran :

- **rewind** : Permite rebobinar la cinta y dejarla en el BOT , para iniciar alguna tarea sobre ella .
- **fsf** : Se desplaza hacia adelante en la cinta , tantas posiciones lógicas se le especifiquen en el contador .
- **bsf** : Se desplaza hacia atrás en la cinta , tantas posiciones lógicas se le especifiquen en el contador .
- **status** : Muestra por pantalla información del estado de la cinta . No es necesario que la unidad tenga una cinta cargada .
- **blksize** : Despliega el tamaño de bloque recomendado para I/O, el cual es usado por tar , cpio , bru , etc . El tamaño máximo, mínimo y actual es reportado y estos pueden ser iguales si el drive no soporta tamaños de bloque varialbles .
- **setblksz** : Define el tamaño del bloque a usarse . La mayoría de drives que soportan tamaños de bloque variables , también soportan el uso de diferentes valores , cuando trabajan en modo de bloque fijo . El tamaño permanece definido hasta que la siguiente cinta sea cambiada , o hasta que el drive se use en modo variable .

Ejemplos :

- Ver el estado de una unidad de cinta :

```
mt -t /dev/rmt/tps0d7 status
```

```
Controller: SCSI
Device: ARCHIVE: Python 01931-XXX5.56
Status: 0x202
Drive type: DAT
Media : Not READY
```

En este caso la cinta no está colocada en la unidad.

- Para grabación de varias cintas lógicas (por ejemplo tres directorios en diferentes cintas lógicas) podriamos realizar el siguiente procedimiento :

Usar el comando para grabar la primera cinta lógica utilizando el dispositivo que no rebobina (la unidad al terminar la grabación, queda situada en el lugar en espera de una orden).

```
# tar cvf /dev/rmt/tps0d7nr  /etc
```

De nuevo usar el comando para grabar la segunda cinta lógica utilizando el dispositivo que no rebobina (la unidad al terminar la grabación, queda situada en el lugar en espera de una orden).

```
# tar cvf /dev/rmt/tps0d7nr  /var
```

Por último grabar la tercera cinta lógica,pero de una vez se le indica a la unidad que rebobine la unidad (utilizando en dispositivo normal que rebobina)

```
# tar cvf /dev/rmt/tps0d7  /usr/lib
```

- Para recorrer y visualizar la información almacenada en una cinta lógia (supongamos que deseamos revisar el contenido de la segunda cinta lógica, de la copia anterior):

Si la cinta esta rebobinada, debemos ubicarnos al principio de lasegunda cinta lógica, usando el comando mt (se debe usar el dispositivo que no rebobina, pues en caso contrario la unidad avanzaría y luego se rebobinaría). Después si podemos listar el contenido de la segunda lógica y si lo deseamos rebobinar indicando el dispositivo que rebobina o quedar situados al principio de la tercera lógica si le indicamos el dispositivo que no rebobina.

```
# mt –t /dev/rmt/tps0d7nr   fsf 1

# tar tvf /dev/rmt/tps0d7nr
```

5.4.PROCEDIMIENTOS PARA DISCOS

En esta sección revisaremos los conceptos necesarios para identificación de dispositivos para discos, particionamiento de los mismos, mantenimiento y monitoreo.

5.4.1.IDENTIFICACIÓN DE LOS DISPOSITIVOS

Esta es la parte que más difiere entre los diferentes sistemas unix y trataremos la forma de indentificación para cada ambiente en su respectiva sección. Ain embargo para generalizar , vamos a exponer un caso.

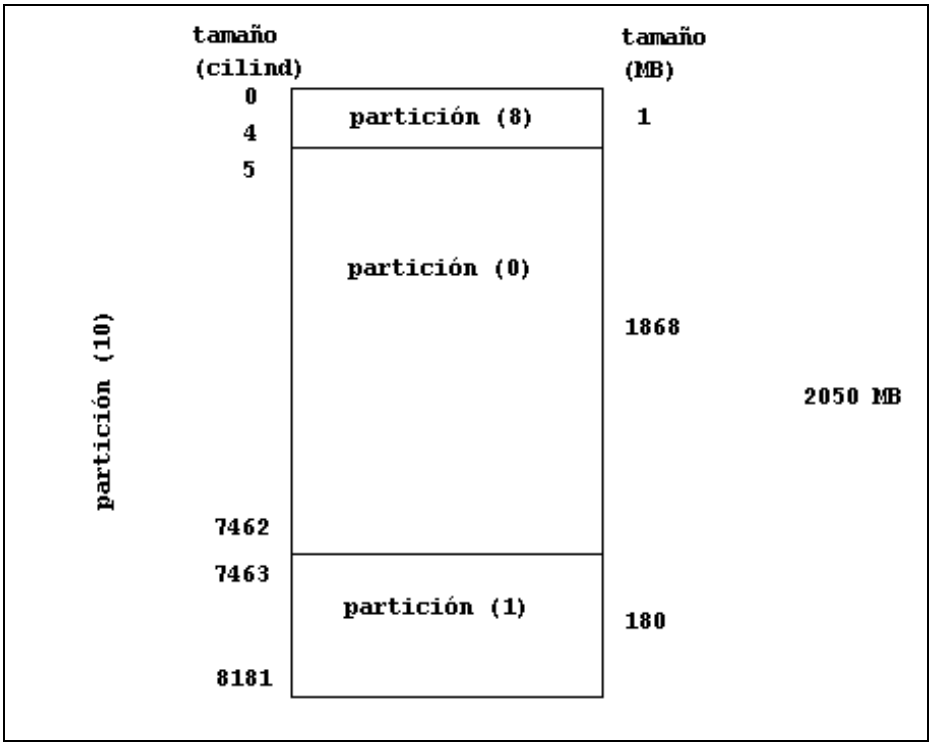
Dependiendo del modelo del disco, se tienen diferentes parámetros como número de cilindros, cabezas, registros por pista, revoluciones por minuto, velocidad de posicionamiento y otros que no tendremos en cuenta, pero que en algún momento puede influir en el rendimiento del mismo.

De esta misma forma, cada disco viene con una tabla de particiones (divisiones lógicas del disco en partes más pequeñas), que podemos modificar y utilizar en la creación de los sistemas de archivos de los usuarios.

Los discos físicos traen unos slices o particiones (divisiones lógicas por cilindros) . Un ejemplo de una tabla de particiones de un disco es el siguiente:

part type	cyls	blocks	Megabytes (base+size)
0: xfs	5 + 7458	2565 + 3826200	1 + 1868
1: raw	7463 + 718	3828765 + 368640	1870 + 180
8: volhdr	0 + 5	0 + 2565	0 + 1
10: volume	0 + 8182	0 + 4197405	0 + 2050

Que equivaldría al siguiente diagrama de partionamiento de un disco de 2GB :



Generalmente las particiones se hacen por cilindros o por megabytes. En cada sistema detallaremos la forma de modificar la tabla de particiones a nuestras necesidades.

Para la identificación del dispositivo que referencia la partición a utilizar para nuestros sistemas de archivos, necesitamos además el número del controlador al que está conectado dicho disco, así como la posición dentro de ese controlador.

Por ahora tenemos que el nombre estaría compuesto de:

- Tres letras que identifican el tipo de controlador al que se esta conectado (Ejm : **dks** , para los dispositivos scsi) .
- Número que identifica la dirección del controlador (Ejm: 0, para representar la dirección cero del controlador scsi).
- La letra “d” seguida de un número que representa la posición del disco en ese controlador (Ejm : **D0,** para representar el primer disco de ese controlador).

- La letra “s” seguida de un número que representa la partición del disco que vamos a tomar (Ejm: **S0,** para representar la partición cero del disco).

Como estos dispositivos están en el directorio /**dev**/**dsk** (listos para ser montados como sistemas de archivos) o /**dev**/**rdsk** (raw devices o crudos), el nombre del dispositivo que identifica el sector a utilizar para creación de un sistema de archivos, podría ser:

/dev/dsk/dks0d0s0

Si ese sector se va a utilizar de una forma cruda (raw), el nombre sería:

/dev/rdsk/dks0d0s0

En algunas versiones nuevas , tanto /dev/dsk y /dev/rdsk, son enlaces a otros directorios.

Si desearamos ver la tabla de particionamiento de un disco, pudiésemos recurrir al comando prtvtoc de la siguiente manera:

```
# prtvtoc /dev/rdsk/dks0d2vol

* /dev/rdsk/dks0d2vol (bootfile "/unix")
*     512 bytes/sector
* Unallocated space:
*       Start        Size
*    17780736         784
*
Partition  Type  Fs   Start: sec    Size: sec   Mount Directory
 0         xfs  yes       4096      3072000   /disco2
 1         raw          3076096       655360
 6         raw          3731456      3512320
 7         raw          7243776      3512320
 8       volhdr               0         4096
10       volume               0     17781520
11         raw          10756096      3512320
12         raw          14268416      3512320
#
```

5.4.2. ADICIONANDO UN NUEVO DISCO

Cuando se adquiere un nuevo disco , para crear nuevos sistemas de archivos , se deben tener en cuenta los siguientes pasos :

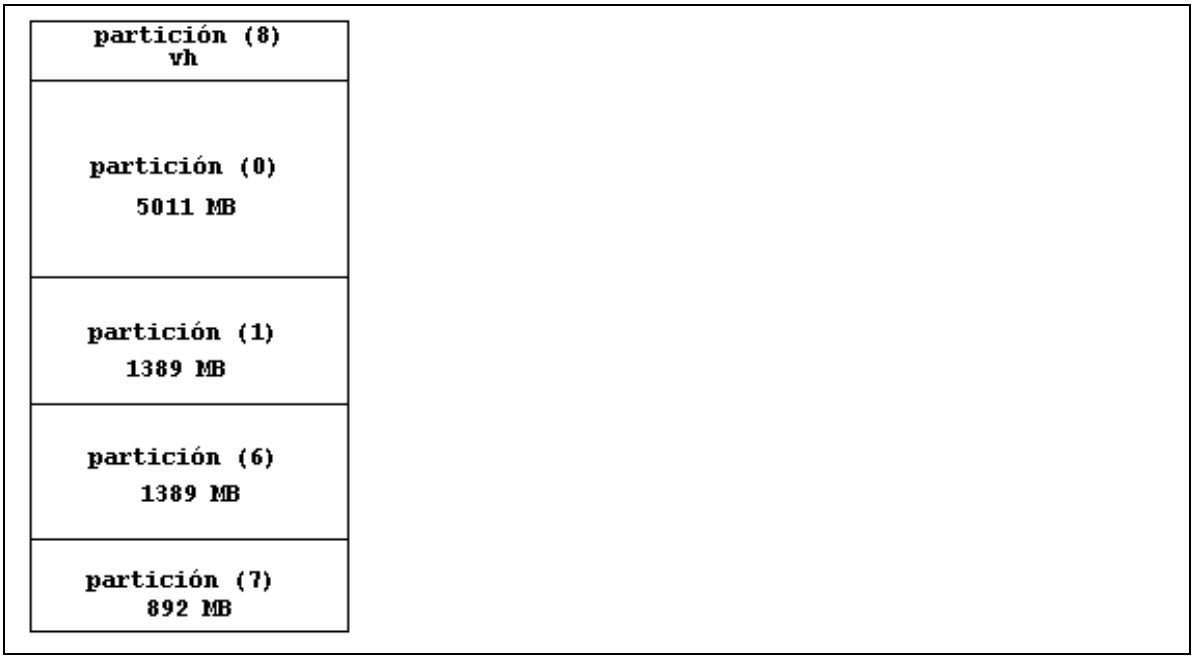
- Instalación física del disco. Se debe elegir el controlador que este menos cargado. En nuestro ejemplo, se instalará un nuevo disco en la dirección siete (7) del controlador scsi 1.

- Después de rebootear, verificar que el sistema detecto el nuevo disco. En Solaris se puede con el comando format y en Iris con el comando hinv. Ejm : en irix

```
# hinv -c disk

Disk drive: unit 7 on SCSI controller 1
Disk drive: unit 1 on SCSI controller 0
Integral SCSI controller 1: Version ADAPTEC 7880
Integral SCSI controller 0: Version ADAPTEC 7880
```

- Particionar el disco de acuerdo a nuestras necesidades. En iris se utiliza el comando **fx**. Este comando sin opciones, invoca un modo de trabajo básico y sencillo. Si se invoca con la opción -x , llama al modo experto del mismo , que nos da más flexibilidad en el trabajo y nos permite más facilidades de trabajo, pero requiere más dominio de los conceptos asociados . En solaris se logra con el comando **format**. La labor de particionar es ajustar los tamaños de las particiones a los requeridos por nosotros, y esto implica indicarle donde empieza la partición y que tamaño tiene. En nuestro ejemplo se va a crear cuatro (4) particiones : de 5011 MB, 1389 Mb, 1389 Mb y 892 Mb ; pero no detallaremos la forma de hacerlo, pues se tratará en las respectivas secciones de los sistemas operativos.

Como lo mencionamos anteriormente, deseamos llegar al siguiente esquema de particionamiento del disco de 9 Gb:




```
----- partitions-----
part  type      cyls            blocks                  Megabytes   (base+size)
  0: xfs        2 + 4330        4740 + 10262100           2 + 5011
  1: xfs        4332 + 1200     10266840 + 2844000       5013 + 1389
  6: xfs        5532 + 1200     13110840 + 2844000       6402 + 1389
  7: xfs        6732 + 771      15954840 + 1827270       7790 + 892
  8: volhdr     0 + 2           0 + 4740                 0 + 2
 10: volume     0 + 7503         0 + 17782110            0 + 8683

capacity is 17783240 blocks
```

- Creación de los sistemas de archivos, en las particiones creadas y que así lo requieran. Si la partición se va a utilizar de una forma cruda (raw), no es necesario realizar los pasos siguientes. Para esto se utiliza el comando mkfs, mkfs_xfs, newfs u otro según el sistema operativo unix. Aquí generalizaremos y utilizaremos el nombre mkfs. Observar que se debe hacer sobe el dispositivo crudo.

```
# mkfs  /dev/rdsk/dks1d7s0
```

```
meta-data=/dev/rdsk/dks1d7s0     isize=256    agcount=8, agsize=160346 blks
data     =                       bsize=4096   blocks=1282762, imaxpct=25
log      =internal log           bsize=4096   blocks=1000
realtime =none                   extsz=65536  blocks=0, rtextents=0

# mkfs  /dev/rdsk/dks1d7s1

meta-data=/dev/rdsk/dks1d7s1     isize=256    agcount=8, agsize=44438 blks
data     =                       bsize=4096   blocks=355500, imaxpct=25
log      =internal log           bsize=4096   blocks=1000
realtime =none                   extsz=65536  blocks=0, rtextents=0

# mkfs  /dev/rdsk/dks1d7s6

meta-data=/dev/rdsk/dks1d7s6     isize=256    agcount=8, agsize=44438 blks
data     =                       bsize=4096   blocks=355500, imaxpct=25
log      =internal log           bsize=4096   blocks=1000
realtime =none                   extsz=65536  blocks=0, rtextents=0

# mkfs  /dev/rdsk/dks1d7s7

meta-data=/dev/rdsk/dks1d7s7     isize=256    agcount=8, agsize=28551 blks
data     =                       bsize=4096   blocks=228408, imaxpct=25
log      =internal log           bsize=4096   blocks=1000
realtime =none                   extsz=65536  blocks=0, rtextents=0
```

El comando mkfs , mkfs _xfs o newfs realiza la creación del sistema de archivos en cada partición de disco a utilizar. En las versiones nuevas, el tamaño del bloque para manejo del sistema de archivos es definido en 4KB (4096 bytes) u 8 Kb, por defecto. En versiones anteriores, era necesario especificarlo en el momento de la creación, pues por defecto lo definia como 512 byte. En esos casos el comando sería :

```
# mkfs -t xfs -d name=dks0d2s7 -b size=4k -l internal,size=2000b
```

- Creación de los puntos de montaje de los sistemas de archivos. Los nombres de los directorios son de libre elección y sólo deben cumplir con las normas de nombres de archivos y directorios en la respectiva versión de unix.

```
# cd /

# mkdir videos job web email
```

- Montar los sistemas de archivos, para el uso de los usuarios.

```
# mount /dev/dsk/dks1d7s0 /videos

# mount /dev/dsk/dks1d7s1 /job

# mount /dev/dsk/dks1d7s6 /web

# mount /dev/dsk/dks1d7s7 /email
```

- Verificar que los sistemas de archivos, son vistos por el sistema.

```
# df -k

Filesystem            Type  kbytes    use       avail     %use Mounted on
/dev/root             xfs   1908820   1765884   142936    93   /
/dev/dsk/dks1d7s0     xfs   5127048   144       5126904   1    videos
/dev/dsk/dks1d7s1     xfs   1418000   144       1417856   1    /job
/dev/dsk/dks1d7s6     xfs   1418000   144       1417856   1    /web
/dev/dsk/dks1d7s7     xfs   909632    144       909488    1     /email
```

- Como este montaje se pierde al rebootear la máquina, se debe incluir en un archivo, para que el montaje sea automático. Este archivo es el /**ect/fstab** en Iris y Linux y el /**etc/vfstab** en Solaris. Se debe editar el archivo e incluir las líneas que hacen referencia a los sistemas de archivos nuevos. En esta línea se especifica: nombre-dispositivo-a-montar punto-de-montaje tipo-de-sistema-de-archivos permisos,nombre-raw-device número-de-secuencia-montaje .

```
# cd /etc
# cat fstab
```

```
/dev/root /  xfs  rw,raw=/dev/rroot 0 0
```

Este es el archivo original de una máquina, donde sólo se esta montando el sistema de archivos del root (/). A este se le adicionan las líneas de los otros sistemas de archivos y debe quedar así:

```
# cat fstab

/dev/root /  xfs  rw,raw=/dev/rroot 0 0
/dev/dsk/dks1d7s0 /videos xfs rw,raw=/dev/rdsk/dks1d7s0 0 0
/dev/dsk/dks1d7s1 /job    xfs rw,raw=/dev/rdsk/dks1d7s1 0 0
/dev/dsk/dks1d7s6 /web    xfs rw,raw=/dev/rdsk/dks1d7s6 0 0
/dev/dsk/dks1d7s7 /email  xfs rw,raw=/dev/rdsk/dks1d7s7 0 0
```

Este archivo, también es leído cuando se invocan las órdenes :

```
# umount -a

# mount -a
```

Las cuales desmontan y montan todos los sistemas de archivos a excepción del root, que figuren en él .

5.4.4. COMANDOS VARIOS ASOCIADOS AL MANTENIMIENTO Y OPERACIONES CON DISCOS Y PARTICIONES

Dentro de estos mencionaremos los comandos que permiten realizar las siguientes tareas:

- **fsck**

Permite chequear y corregir errores en los sistemas de archivos .

Sintaxis :

```
fsck –y /dev/rdsk/dispositivo
```

La opción –y , responde automáticamente con yes a todas las preguntas. Puede no ser conveniente en algunos casos. La partición debe estar demontada para su chequeo. Si se da sin opciones corre un fsck a todas las particiones que lo requieran y que estén listadas en el /etc/fstab o /etc/vfstab.

5.5. MANEJO DE QUOTAS DE DISCO

Cuando un usuario tiene asignado un directorio dentro de una partición de disco, este puede escribir en ella hasta que la partición se llene, perjudicando a los demás usuarios y procesos que residen en esa partición. Es conveniente en algunas circunstancias asignar limites de consumo de espacio en disco dentro de una partición a un usuario y esto es llamado quota de disco. El proceso para asignar una quota de disco a un usuario es:

- Activar el uso de quotas en una partición. Para esto se el administrador usa el comando quotaon . Se debe tener creado en la raíz de esa partición un archivo llamado quotas y que el propietario sea root.

```
# df -k
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/dsk/c0d0s0      1558350  928260  567756    63%    /
/dev/dsk/c0d0p0:boot   11484    1508    9976    14%    /boot
/proc                      0       0       0     0%    /proc
mnttab                     0       0       0     0%    /etc/mnttab
fd                         0       0       0     0%    /dev/fd
swap                  681952      84  681868     1%    /var/run
swap                  682172     304  681868     1%    /tmp
/dev/dsk/c0d0s7     17478129    1119 17302229     1%    /export/home

# cd /export/home
# touch quotas
# ls -l
total 26
drwxr-xr-x   2 root     other        512 May 12 17:02 apagado
drwxr-xr-x   2 104      other        512 May 12 17:36 apagar
drwx------   2 root     root        8192 Mar 20 11:21 lost+found
drwxr-xr-x   2 operador other        512 May 12 18:00 operador
-rw-r--r--   1 root     other          0 May 19 16:39 quotas
```

```
drwxr-xr-x   2 usercsh  staff        512 Apr  9 16:37 usercsh
drwxr-xr-x   2 usersh   staff        512 Apr  9 17:23 usersh

# quotaon -v /export/home
/export/home: quotas turned on
```

- Se define a que usuario y que quota se le especifica. Se invoca el comando edquota y el nombre del usuario (solo lo puede hacer root):

```
#edquota operador
```

Esto invoca al editor vi o al definido en la variable EDITOR y se debe incluir la siguiente línea:

```
fs /export/home blocks (soft = 2, hard = 4) inodes (soft = 0, hard = 0)
```

Allí se le define de quota al usuario operador 2 bloques de disco (de 1024 bytes) en su límite soft. Este soft es como una primera advertencia o limite de espacio; al llegar a el el usuario puede sobrepasarse durante un periodo de tiempo (manejado por el timer) , que pueden ser dias , pero nunca puede sobrepasar el limite hard.

- Chequear la quota (quotacheck)

```
bash-2.05# quotacheck -v /export/home
*** Checking quotas for /dev/rdsk/c0d0s7 (/export/home)
operador   fixed:  files 0 -> 2  blocks 0 -> 4
bash-2.05#
```

- Pruebas. Se logea como el usuario operador y se trata de copiar un archivo que exceda su quota:

```
bash-2.05# su - operador
Over disk quota on /export/home, remove 0K within 7.0 days
Sun Microsystems Inc.   SunOS 5.9       Generic_112234-03       November 2002
Usuario Impresora>id
uid=200(operador) gid=1(other)
Usuario Impresora>cp /etc/inittab ./
```

```
quota_ufs: over hard disk limit (pid 1231, uid 200, inum 21, fs /export/home)
cp: ./inittab: Disc quota exceeded
Usuario Impresora>ls -l
total 0
Usuario Impresora>
```

5.6. TALLER EVALUATIVO

Hacer un menú para las siguientes tareas:

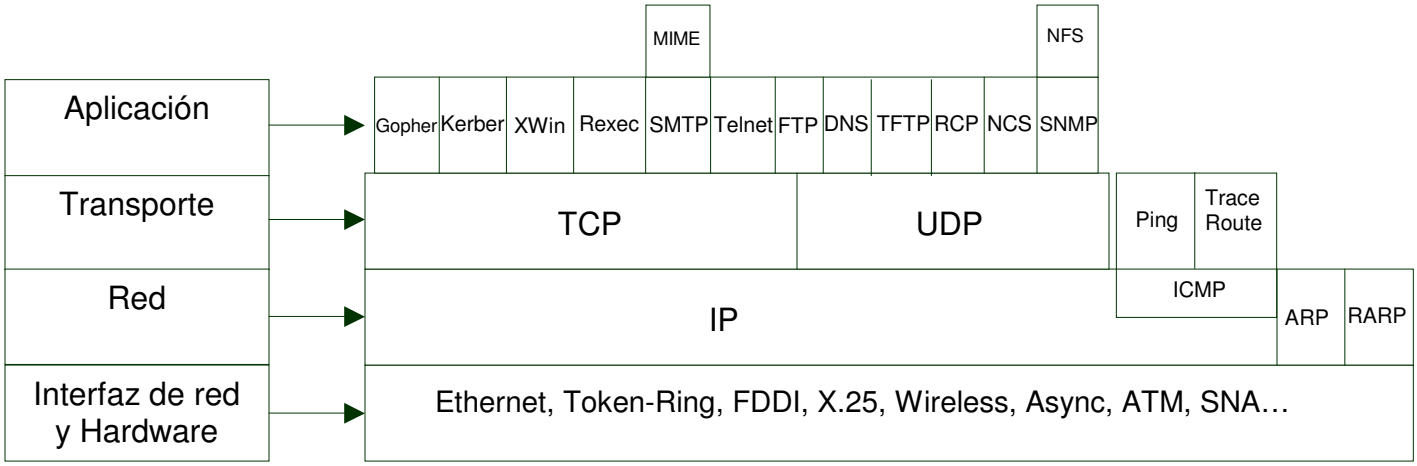
- REPORTE DE PROCESOS QUE MAS CONSUMEN RECURSOS (CPU Y MEMORIA)
- REPORTE DE SISTEMAS DE ARCHIVOS CON USO SUPERIOR AL 90%
- CUANTIFICAR ESPACIO USADO POR UN USUARIO EN UNA PARTICION
- BUSQUEDA Y LIMPIEZA DE ARCHIVOS NO DESEADOS
- BUSQUEDA DE ARCHIVOS POR DIFERENTES CRITERIOS
- MANEJO BASICO DE BACKUPS
- SALIR

6. CONCEPTOS DE REDES

6.1. TCP/IP , ARQUITECTURA ISO EN EQUIPOS UNIX

TCP/IP es un producto de comunicaciones que permite conectar Host que corran el Transmission Control Protocol (TCP) y el Inter Protocol (IP) sobre una red Local, fue desarrollado por el Departamento de defensa de los Estados Unidos para uso militar y adaptado como un estándar para comunicaciones a través de redes LAN.

Su arquitectura según el modelo ISO/OSI, se define así:



Inicialmente, los dos primeros niveles (1 y 2 que corresponden a Nivel Físico y de enlace respectivamente), se declararon indefinidos, para dejar una elección libre del medio de transmisión de datos, sin que esto afecte los demás niveles ni las características del TCP/IP.

En el caso de las redes de familia ethernet, podemos encontrar lo siguiente con respecto a esos niveles:

Ethernet: Fue de los pioneros en redes LANs corriendo TCP/IP. Actualmente existen también las implementaciones de FastEthernet y Gigabitethernet.

Características:

- Rata de transferencia de datos alta (10 Mb/seg, pero ya existen 100 y 1000 Mbps))
- Baja rata de errores, ofrece detección más no corrección.
- Protocolos no orientados a conexión, no garantizan la entrega de los frames.
- No utilizan buffers o control de flujo cuando envían mensajes.

Ethernet era menos costosa para implementar en hardware que otros protocolos debido a la simpleza de sus interfaces.

Es ineficiente con cargas sobre el 40%. Por el método de acceso que utiliza (CSMA/CD), pero veremos más adelante cuales son las tendencias de las redes actuales.

6.1.1. ANÁLISIS DEL FRAME ETHERNET

Un frame es una unidad simple de datos transportada a través de la LAN. La información contenida en el frame es análoga a la información requerida para enviar y registrar un paquete por el correo, tal como :

- Dirección del destino
- Dirección del remitente
- Recibo de entrega
- Peso
- Contenido del paquete

Los host en una red LAN ethernet usan esta información para recibir y transmitir datos.

Preamble: Es usado para sincronizar el interface entre los hosts que reciben y transmiten.

Dirección destino: Es la dirección única de hardware del host destino. Una dirección ethernet está compuesta de 48 bit y es de la forma : ejm . 8:0:20:e:56:7d
Los primeros tres campos son reservados para el fabricante y son diseñados por el IEEE (Institute of Electronic and Electrical Enginners).

Dirección origen: Es la dirección única de hardware del host que envía.

Type/logitud: La información que provee esta basada en la revisión de ethernet (V1,V2,802.3) usada.

Datos: Tiene un mínimo de 46 bytes, hasta un máximo de 1500 bytes.

CRC : Cyclical Redundandy Check, es usado para detección de errores.

6.1.2. MÉTODO DE ACCESO

Los host envían mensajes sobre una red LAN ethernet usando el protocolo de nivel de Enlace CSMA/CD (Carrier Sense Multiple Access with Collision Detectet).

Este permite que todos los dispositivos se comuniquen sobre un medio sencillo, que sólo uno puede transmitir al tiempo, y que todos pueden recibir simultáneamente. Si dos hosts transmiten en el mismo instante, la colisión en la transmisión es detectada y ambos dispositivos esperan un tiempo aleatorio antes de volver a intentar.

Mensajes unicast: Un host envía un mensaje a otro usando la dirección unicast (la propia del host).

Mensajes broadcast: Un host envía un mensaje a todos los hosts en la red ethernet usando la dirección broadcast (ff:ff:ff:ff:ff:ff)

Para comprender los otros niveles del Modelo TCP/IP, necesitamos los siguientes conceptos:

- Internet Protocol (IP): Es un protocolo que provee un servicio de Datagrama, que organiza los datos en paquetes, enruta estos y reensambla los datos al llegar a su destino. En un servicio de datagrama los mensajes son pasados al host en el orden de llegada sin chequeo de errores. Cada datagrama es tratado como una unidad y necesita una dirección en cada paquete.
- Transmission Control Protocol (TCP) : Es un protocolo a nivel de transporte, encargado de realizar las conexiones entre host. Registra la secuencia de los paquetes, realiza transmisiones y otros.

- User Datagram Protocol (UDP) : Es usado para evitar y recibir información entre host y soportar transferencia de archivos por TFTP. Soporta la transmisión de segmentos de datos sin secuencia.
- TELNET: Este servicio permite conectar dos sistemas, emulando las terminales propias de cada uno.
- File Transfer Protocol (FTP) : Transferencia de archivos entre dos computadores.
- Simple Mail Transfer Protocol (SMTP): Transferencia de correo.
- Trivial File Transfer Protocol (TFTP): Permite intercambiar archivos, pero no es tan completo como el FTP.

En ambiente de trabajo TCP/IP, las estaciones, servidores y otros elementos se comunican entre sí usando una única dirección lógica de 32 bit. Si los nombres son usados en vez de direcciones, estos deben ser traducidos a una dirección numérica antes de que el tráfico pueda ser liberado.

Cada compañía listada en internet es vista como una red simple que debe ser alcanzada antes de que un host individual de esta pueda ser contactado. Cada compañía tiene una dirección.

6.1.3. DIRECCIONES IP

Una dirección IP esta compuesta de 32 bits y consta de dos partes:
- Número de Red
- Número de Hos

El formato de la dirección es conocido en forma decimal:

- Ejemplo de dirección: 131.108.122.204
- Cada bit en el octeto tiene un peso binario, tal como (128,...,4,2,1)
- El mínimo valor de un octeto es cero (0), si contiene todos los bit en cero.
- El máximo valor del octeto es 255, si contiene todos los bit en uno (1).

Los números de red son administrados por Internet Network Information Center (InterNIC).

Para una administración más fácil los números de red se dividieron en clases. Los bits más significantes de cada dirección determinan la clase de esta:

- Clase A: Rango de direcciones desde 1.0.0.0 hasta 126.0.0.0, para un número de posibles direcciones de host de 16,777,214.
- Clase B: Rango de direcciones desde 128.1.0.0 hasta 191.254.0.0, para un número de posibles direcciones de host de 65,534.
- Clase C: Rango de direcciones desde 192.0.1.0 hasta 223.255.254.0, para un número de posibles direcciones de host de 254.

Las clases A, B, C son las más comunes, aunque también hay definidas clases D y E.

Para reconocer la clase de dirección y poder tomar decisiones de enrutamiento, muchos enrutadores aplican la regla del primer octeto:

BIT DE MAYOR ORDEN	OCTETO EN DECIMAL	CLASE DE DIRECCION
0	1 - 126	A
10	128 - 191	B
110	192 - 223	C

DIRECCIONES DE HOST

Cada dispositivo o interface debe tener un número de host diferente de cero (0). Una dirección de host con todos los bits en uno (1) es reservada para efectos de broadcast en la red. Un dirección de host de cero (0) representa la red donde estamos ubicados. Las tablas de enrutamiento contienen generalmente direcciones de red y no de host.

Para facilitar administración y mejorar la eficiencia , se pueden dividir las redes en subredes. Desde el punto de vista de direcciones las subredes son una extensión del número de red. El administrador de la red debe decidir el tamaño de las subredes. Para poder entender estas subredes se necesitan equipos de red tales como enrutadores o una máquina que actúe como tal.

Cuando se manejan subredes se debe entender y manejar la máscara conocida como subnet mask. Esta lo que indica es cuales bits en el campo del host son usados para especificar las diferentes partes (subredes) de una red en particular.

La subnet mask tiene también 32 bit de tamaño y esta escrita en cuatro (4) octetos. En estos pueden ir los siguientes valores:

- Binario 1 para bit de red
- Binario 1 para bit de subred
- Binario 0 para bit de host

Ejemplo:

Dirección IP	131	108	2	200
Subnet Mask por defecto	255	255	0	0
Subnet Mask de 8 bit	255	255	255	0

Usa los bits del host, empezando en los bit de más alta posición.

128	64	32	16	8	4	2	1		
1	0	0	0	0	0	0	0	=	128
1	1	0	0	0	0	0	0	=	192

1	1	1	0	0	0	0	0	=	224
1	1	1	1	0	0	0	0	=	240
1	1	1	1	1	0	0	0	=	248
1	1	1	1	1	1	0	0	=	252
1	1	1	1	1	1	1	0	=	254
1	1	1	1	1	1	1	1	=	255

Subred Mask sin subredes:

	red		host	
131.108.2.160	10000011	01101100	00000010	10100000
255.255.0.0	11111111	11111111	00000000	00000000
	10000011	01101100	00000000	00000000
	131	108	0	0

Subred Mask con subredes:

	red		subred	host
131.108.2.160	10000011	01101100	00000010	10100000
255.255.255.0	11111111	11111111	11111111	00000000
	10000011	01101100	00000010	00000000
	131	108	2	0

El número de red es extendido en 8 bit más.

Ejemplo :

Internic asignó una direción clase C de 201.222.5.0 . Asumir que se necesitan 20 subredes, con 5 host cada una. Nosotros necesitamos subdividir el último

octeto en una subred y una porción para los host y determinar la máscara correcta a usar para lograr esto.

Seleccionando 5 bit mask, se permiten las 20 subredes. Cada dirección de la subred es múltiplo de 8, así : 201.22.5.16, 201.222.5.32, 201.222.5.48, etc.

Los bits restantes del último octeto son usados para representar el host. Los 3 bit pueden representar los 5 host necesitados. Los números de los host, serán así : 1,2,4, etc.

```
0   0   0   0   1 0   0   1
----------------- ---------
subred            host
```

La primera subred sería la 8 (pero no se utiliza al igual que la última). La primera dirección de host sería el 9.

En el campo de la subred , tanto como en el del host, no pueden ir todos los números en 0 (cero) o 1 (uno), pues estas direcciones tienen un significado especial.

La dirección final de los hosts es una combinación de la subred más el número del host. Así los host de la subred 201.222.5.16 estan en el rango de 201.222.5.17 hasta 201.222.5.22.

La subnet mask utilizada sería : 255.255.255.248

Si vamos a analizar la dirección IP 201.222.5.121 encontraríamos :

	red			Subn+host
201.222.5.121	11001001	11011110	00000101	01111 001
255.255.255.248	11111111	11111111	11111111	11111 000
Subnet	11001001	11011110	00000101	01111 000
	201	222	5	120

Subnet Address = 201.222.5.120
Host Address = 201.222.5.121 - 201.222.5.126

Broadcast Address = 201.222.5.127

Existe una tabla que nos puede ayudar en esta planeación de subredes, para la clase C :

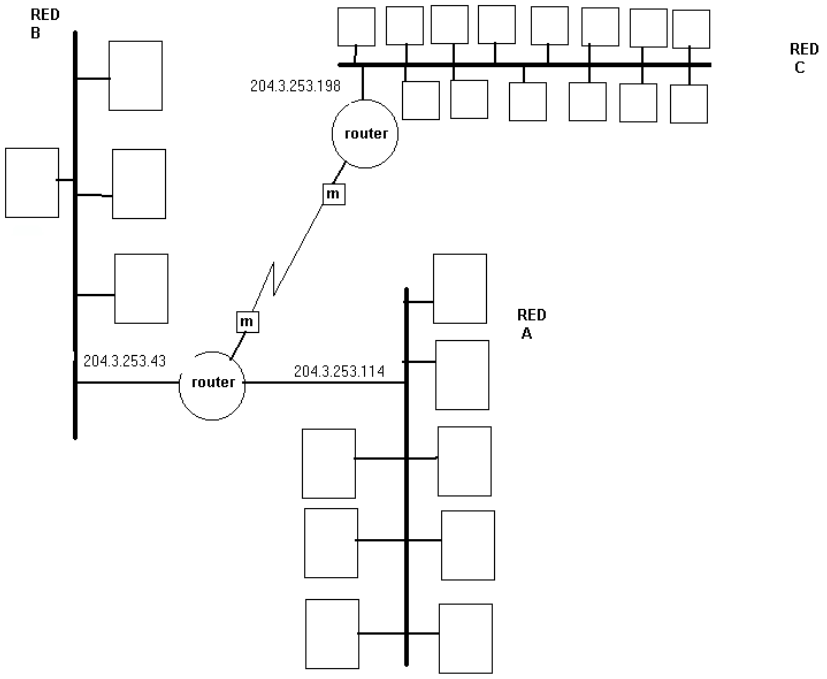
#BITS	SUBNET MASK	#SUBNETS	#HOSTS
2	255.255.255.192	2	62
3	255.255.255.224	6	30
4	255.255.255.240	14	14
5	255.255.255.248	30	6
6	255.255.255.252	62	2

6.1.4. EJERCICIO DE ASIGNACIÓN DE IPS

Una empresa cuenta con el siguiente grupo de direcciones, asignadas por su proveedor : 204.2.253.0 mask 255.255.255.0.

Esta máscara le permite contar con 254 IP para sus equipos. La empresa tiene 3 sucursales y en cada una de ellas tiene una infraestructura LAN. Desea repartir las IP asignadas, en sus sucursales, de acuerdo a las necesidades y cantidades de equipos que las requieren.

En la siguiente figura , hacer el esquema de cómo quedarían las posibles subredes, si ya se han asignado unas IP a algunos equipos.



El ejercicio consta de tres partes :

- Se deben asignar las direcciones para las subredes entre los enlaces WAN, utilizando direcciones privadas.
- Se deben asignar las subredes para las tres sedes partiendo de que no se tiene ninguna IP asignada a equipos de las LAN
- Se deben asignar las subredes para las tres sedes partiendo de que ya se tienen algunas IP asignadas a equipos de las LAN.

6.2. DISPOSITIVOS ACTIVOS EN REDES

Dentro del cableado estructurado existen una serie de dispositivos que clasificamos como activos y pasivos. Cada red de datos cuenta con dos tipos de

equipos requeridos, los activos o aparatos que manipulan y manejan datos, como enrutadores, concentradores y switches, y los pasivos o que simplemente permiten que la información fluya a través de ellos, como cableado, conectores, etc.

6.2.1. HUB[2]

El hub era el dispositivo más importante de todas estas redes, ya que al contrario de lo que sucedía con las redes que emplean cable coaxial, donde el mismo iba de computadora a computadora, en las redes con cable UTP el cable va de cada una de las computadoras hacia al hub necesariamente. Esto le da a la red una topología física, netamente en estrella, aunque la transmisión interna sea en bus por difusión (por tal razón a los hubs también se les conoce como repetidores).

El hub es simplemente un dispositivo que trabaja en la capa física de las redes, y tiene por objeto repetir la señal que proviene de una de sus entradas hacia absolutamente todas las otras. En este proceso el hub puede, según sus características particulares, mejorar la señal ampliándola, reajustando los bits, etc. En síntesis, realizando el proceso de regeneración digital de la señal.

Como se mencionó en una sección anterior la tendencia es que estos elementos sean reemplazados por switchs.

El clásico modelo de una red UTP categoría 5 era el que se muestra en la figura, y un hub más específicamente se muestra en la figura siguiente.

El hub se constituía como el centro de toda la red UTP, y al mismo se conectan tanto terminales como servidores. Actualmente esta función la cumplen los switch.

[2] Medios de Transmisión . David Redondo. www.aebius.com

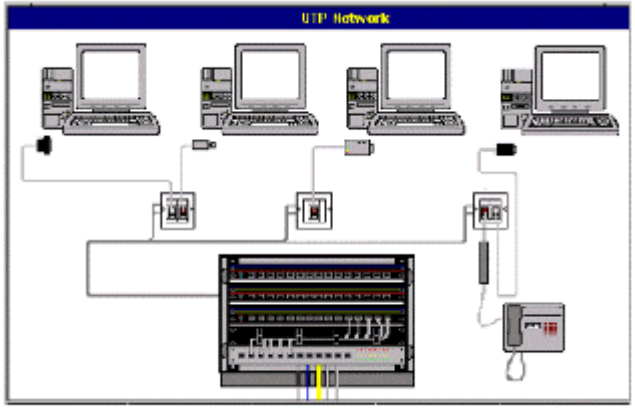


Fig 4 Esquema típico de una red UTP

Al adquirir un hub deben intervenir algunos puntos y criterios importantes.

- Los conectores para los cables UTP se pueden hallar en la parte anterior como en la parte posterior del hub, y existen modelos que soportan cualquiera de las modalidades. Este punto debe ser discernido por el administrador de la red, de acuerdo a sus requerimientos particulares de ambiente. Existen Hub con diferentes números de puertos (8,12, 16, 24 puertos)

Fig 5 Vista anterior y posterior de un hub

- Otro punto importante que debe cumplir un hub es el de poder trabajar tanto con comunicaciones de 10Mbps como de 100Mbps, esto con el objeto de permitir migrar fácilmente redes de 10Mbps hacia 100Mbps sin tener que emplear dispositivos diferentes para cada una de las mismas. Por supuesto, el hub realiza todas las tareas de buffering o control de flujo entre ambas velocidades.

- Cuando se adquiere un hub este tiene una determinada cantidad de puertos disponibles, la misma que por un proceso de crecimiento de la red puede quedar insuficiente, por esta razón, el hub debe soportar conexiones en pila, es decir, poder emplear un puerto para unirse a otro hub ampliando de esta forma la cantidad de puertos disponibles. Esto no es muy aconsejable.

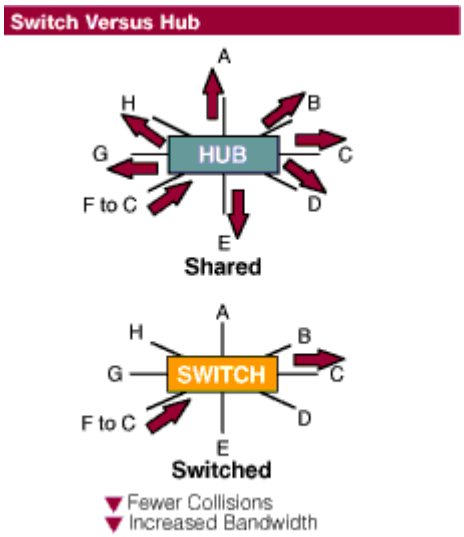
- Normalmente, y dentro el denominado cableado estructurado, el hub conecta elementos de un piso de un edificio con un medio de transmisión denominado vertical o principal o backbone, que bien puede ser fibra óptica. En este sentido el hub debe poseer la capacidad de soportar

diversos tipos de backbone, particularmente el de fibra óptica, no solamente en el caso de tratarse de un edificio, sino también en un campus universitario u otro tipo de ubicación de una institución.

- Finalmente, si se desea tener un software de gestión en la red, es aconsejable que estos elementos se dejen monitorear.

6.2.2. SWITCH (CONMUTADOR)

Con la conmutación decimos adiós a la red compartida. A diferencia del Hub que difunde los paquetes a toda la red y por lo tanto sólo permite que haya una terminal enviando/recibiendo datos, el Conmutador o Switch es más inteligente y envía el paquete sólo a su destinatario, y por lo tanto, pueden existir varios usuarios enviándose paquetes siempre y cuando el destinatario (no la red) esté libre.



Así con el switch se garantiza la velocidad ofrecida entre los equipos que se comunican (10, 100, 1000 Mbps).

En primer lugar existen diferentes tipos de switchs, en función de su inteligencia (la capa en donde trabajen) y de su costo. Los switch de capa 2 toman decisiones de envío basándose en la dirección física o MAC de destino contenida en cada paquete. Al contrario que el bridge, el switch puede reenviar los datos con

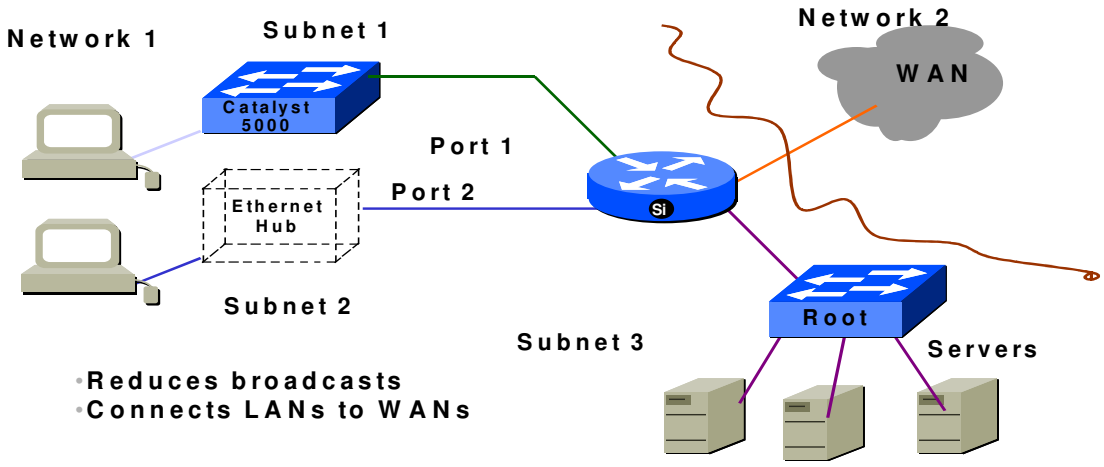
periodos de latencia muy bajos. Existen dos tipos de switch de capa 2 según el modo de reenvío:

- Cut-Thorugh: los conmutadores inician el proceso de reenvío antes de haber recibido la trama completada (no comprueba la integridad de la trama).
- Store-and-forward: que leen y validan el paquete entero, incrementando la latencia proporcionalmente al tamaño del paquete, pero ganando control de flujo. En definitiva, los entornos más beneficiados son aquellas redes LAN complejas que en la actualidad utilizan routers para su segmentación.

Los switches capa 3 permiten hacer enrutamiento Lan-LAN.

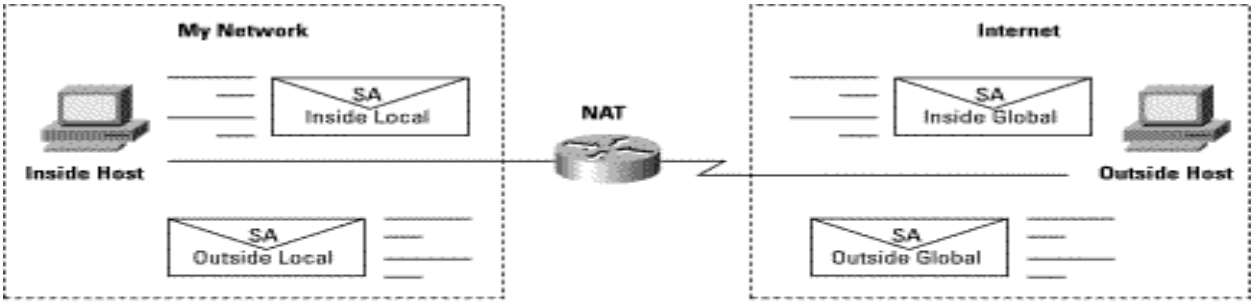
6.2.3. ROUTER

Los router son aún más inteligentes que los switch (capa 2). A diferencia de éstos, que sólo leen la dirección de Control de Acceso al Medio (MAC), que es el identificador del hardware de un dispositivo de red, los router analizan la información contenida dentro de un paquete de red. Los router leen cada paquete y lo envían a través de la ruta más eficiente hacia su destino, de acuerdo con un conjunto de reglas. Sin embargo, esta inteligencia requiere más procesamiento, por lo cual los router son menos eficientes que los switch. Los router a menudo son utilizados para conectar redes que están separadas geográficamente (LAN-WAN), para lo cual se emplea tecnología con baja velocidad como líneas RDSI o T1, pares telefónicos dedicados, etc.



Estos dispositivos no solo sirven para este tipo de labores, ya que pueden prestar servicios adicionales como estos entre otros :

- Autenticación de usuarios. Los dispositivos que soportan líneas asincrónicas para permitir u otras que permitan acceso conmutado a una red, se pueden configurar para que autentiquen los usuarios que deseen ingresar a la red, bien sea con usuarios creados en el dispositivo o validando usuarios en un servidor (radius, tacacs).
- Lista de control de acceso. Permitir creación de listas donde se pueda restringir servicios.
- Network Address Translation (NAT).Opera sobre un router conectando dos redes juntas. Una de estas redes, llamada interna, usa direcciones obsoletas o privadas que necesitan ser convertidas en direcciones legales o públicas antes de que los paquetes sean pasados a la red externa.



An IP Address is Either Local or Global
Local IP Addresses are Seen in the
Inside Network
Global IP Addresses are Seen in the Outside Network

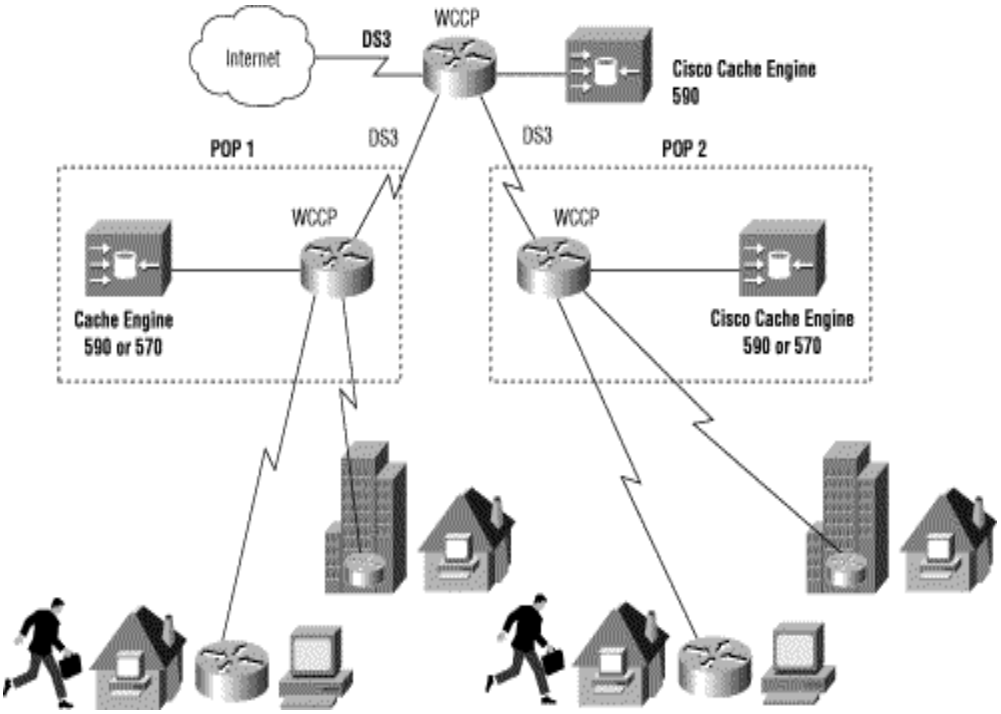
6.2.4. FIREWALL

Un firewall es un sistema o grupo de sistemas que aseguran unas políticas de control de acceso entre dos redes. Se basan en mecanismos básicos de bloquear trafico o permitirlo. Algunos hacen más énfasis en bloquear mientras que otros en permitir. Realmente lo más sobresaliente es que implementan políticas de control de acceso.

Internet como cualquier sociedad esta plagada con tipos de personas que disfrutan, si lo comparamos con sociedad, pintando en los muros de otras personas, abriendo su correo, etc. El propósito del firewall es mantener estas personas fuera de nuestra red.

Algunos firewall permiten sólo el tráfico email a través de ellos, otros se centran en proteger servicios con problemas conocidos. Otros firewall bloquean tráfico de un lado de la red ,pero permiten a los usuarios internos comunicarse hacia fuera de la misma. También son importantes, porque proveen un punto de choque, donde la seguridad y la auditoria puede ser impuestas.

6.2.5. DISPOSITIVOS DE CACHÉ



Ofrecen servicios como :

Descongestión de ancho de banda de canales WAN, y altos costos en la transmisión.
Mejorar la calidad del servicio

Maximizar y controlar la disponibilidad de los contenidos de internet que son vistos por los clients.

En el ejemplo anterior, el dispositivo de caché llamado Cisco Cache Engine 500 series acelera la entrega de contenido, optimiza e la utilización de ancho de banda y puede controlar el acceso a internet.

6.3. ARCHIVOS Y COMANDOS INVOLUCRADOS

6.3.1. ARCHIVOS DE CONFIGURACIÓN

El papel del administrador es configurar la red, creando o modificando los archivos que mencionaremos a continuación:

/etc/hosts Contiene los nombres y las direcciones Internet de los Hosts involucrados. Para cada host se debe dar:

- Dirección internet.
- Nombre oficial del host.
- Alias o nombre con el que vamos a referenciar el host. El alias para el host local es seguido de la palabra "Localhost".Esta lista no debe incluir todas las máquinas, si se esta usando DNS.

/etc/hosts.equiv Permite a los usuarios de sistemas remotos entrar al sistema sin dar password, si se cumple:

- El sistema remoto tiene una entrada en el hosts.equiv.
- El usuario remoto tiene un user-id en el host local.

El remote login (rlogin) usa este archivo para autenticar usuarios.Esto afecta a todos los usarios de la máquina. Si se desea personlizar se debe recurrir al siguiente archivo.

.rhost Permite a usuarios individuales de sistemas remotos, entrar al sistema sin dar password, cuando el host remoto no tiene una entrada en el archivo hosts.equiv. El usuario necesita colocar

un .rhosts en su directorio de trabajo. Este archivo es usado por el "rlogin -l " y otros comandos. El formato del archivo es: (por cada línea)

hostsname user-id

Donde hostname es el nombre del host que se desea accesar y el respectivo nombre de usuario para ese host.

Existen otros archivos que el sistema crea automáticamente, y se pueden observar; que detallaremos a continuación:

/etc/protocols Protocolos usados por aplicaciones corriendo sobre TCP/IP.

/etc/services Servicios disponibles.

/etc/inetd.conf Especifica los procesos y demonios que arrancaran con la red.

/etc/resolv.conf Configuración de DNS. Especifica el dominio y la máquina que actua con servidor de nombres. Además el órden en que sé resolveran los nombres. Ejemplo:

```
domain     psa.edu.co
nameserver 207.24.18.5
```

6.3.2. COMANDOS INVOLUCRADOS.

Si se desea verificar que la dirección IP y otras caracteristicas fueron definidas apropiadamente para la tarjeta de red, se puede usar los comandos:

```
#netstat -i

Name Mtu   Network       Address            Ipkts Ierrs    Opkts Oerrs  Coll
```

```
ef0  1500  200.25.18      pelicano       13033995  1302   964346    39 32492
lo0  8304  loopback      localhost       1065823     0  1065823     0     0


#ifconfig ef0
ef0:
flags=1c63<UP,BROADCAST,NOTRAILERS,RUNNING,FILTMULTI,MULTICAST
,CKSUM>
        inet 200.25.18.4 netmask 0xffffffc0 broadcast 200.25.18.63
```

Con el netstat se puede observar el nombre de la interface de red. En este caso se llama ef0. Además información sobre paquetes entrantes, salientes y colisiones.

Con el comando ifconfig, se puede observar y modificar los parámetros cargados a las interfaces de red.

El comando netstat tiene otras opciones:

netstat [-a] Muestra información de todos los servers activos.

netstat [-n] Muestra las direcciones internet .

netstat [-p] prot Da información solo del protocolo especificado . Este puede ser : tcp , udp o ip .

Existe un comando que permite revisar la conexión de los hosts, enviando una serie de caracteres al otro equipo y esperando recibir estos de nuevo. Es el comando "ping", cuyo formato es:

ping nombrehost

Si solo se da el nombre del host, se envía una paquete de caracteres y se espera recibir estos nuevamente, en forma indefinida hasta que el usuario detenga la emisión. Se presentan unas estadísticas sobre los caracteres perdidos para estudiar el estado de la conexión y del medio físico. Se puede definir el tamaño del paquete enviado y la cantidad de veces que se realiza esta operación.

6.3.3. COMUNICACIÓN CON OTROS EQUIPOS UNIX

Los equipos con sistema operacional UNIX,conectados por medio de TCP/IP, soportan además , las llamadas utilidades Berkeley (utilities BSD) :

- **Remote Copy (RCP):** Para copiar tanto archivos como directorios entre dos hosts en red.

  Para copiar un solo archivo:

  rcp [nodofte:archfte] [nododest:archdest]

  Para copiar varios archivos:

  rcp [nodofte:]arch1 ...[nodofte:]archn [nododest:]directorio

  Para copia de directorios:

  rcp -r [nodofte:]direcfte [nododest:]direcdest

- **Remote Login (rlogin):** Permite hacer login en el host remoto. Se puede tener:

  Login bajo el mismo user-id : entrar al host remoto con el mismo user-id con el que trabajamos en el host local. El comando es:

  rlogin nombre-host-remoto

  Login con diferente usuario ; el comando es :

  rlogin nombre-host-remoto -l user-id-remoto

- **Remote Shell (remsh):** Permite correr comandos del host remoto. Se pueden tener varias combinaciones, como:

  Ejecutar un comando:

  remsh nombre-host-remoto comando

Ejecutar proceso local y remoto:

comando-local | remsh nombre-host-remoto comando-remoto

Ejecutar comando remoto y local:

remsh nombre-host-remoto comando-remoto | comando-local

6.3.4. COMUNICACIONES CON OTROS SISTEMAS OPERACIONALES

Se puede comunicar con otros equipos de diferente sistema operacional, que corran TCP/IP.Los servicios prestados son:

- **TELNET :** Permite hacer login en un host remoto.

  La sintaxis del comando es la siguiente:

  Telnet [nombre-host]

  Si solo Telnet es digitado, el sistema lo situara en modo comandos, cuyo prompt es " Telnet> ". En este modo se pueden dar los siguientes subcomandos:

Open nombre-host	Para conectarse a un sistema.
close	Cierra la conexión.
quit	Salir de TELNET.
help	Ayuda .
status	Muestra información de la conexión actual .

  En el momento que la conexión es hecha, se pasa a modo de transferencia de datos y se interactua con el hosts remoto para realizar Login.

- **FILE TRANSFER PROTOCOL (FTP):** FTP permite realizar transferencia de archivos EXL y otros equipos que corran TCP/IP. La sintaxis del comando es:

  ftp [opciones] [nombre-host]

  donde la opciones pueden ser:

  -d Habilita debug

  -g Deshabilita el globbing (uso de wildcards)

  -i Habilita prompt interactivos, es decir,pedir confirmación para cada archivo al trabajar con wildcars.

  -n No pregunta por un usuario al establecer la conexión.

  Este comando nos coloca en un subsistema cuyo prompt es "ftp>", en donde disponemos de los siguientes subcomandos:

  ! [comando] Ejecutar comando del sistema local . Sale al prompt de UNIX, donde se regresa con "exit".

  append arch [arch-rem] Agrega el contenido del archivo local (arch), al archivo remoto (arch-rem).

  type ascii Pasa el tipo de archivo a ASCII.

  type binary Pasa el tipo de archivo a IMAGE.

  type image Pasa el tipo de archivo a IMAGE.

  bell Activa o desactiva sonido de campana al terminar copia de archivos.

  bye Termina FTP y la sesión remota.

  cd directorio Cambia el directorio de trabajo del host remoto .

close	Termina sesión remota y permanece en FTP.
debug [on/off]	Al activar debug (on), cada comando enviado al sistema remoto es mostrado precedido por el string -->
delete archivo	Borra archivo en el host remoto.
dir [direct] [archivo]	Lista el contenido del directorio del host remoto y sitúa este en el archivo del host local .
get arch-rem [arch-local]	Transfiere archivo del host remoto al local.
glob	Interpretación de wildcards. Si esta en "on", se pueden usar estos; si es "off", los wildcars son interpretados literalmente .
hash	Impresión del símbolo "#", por cada bloque de datos transmitido (1024 bytes).
lcd directorio	Cambia el directorio de trabajo del host local.
ls [directorio [archivo]]	Muestra información abreviada del contenido del directorio remoto y sitúa esta en un archivo del host local.
mdelete archivos	Borra archivos del host remoto.
mkdir directorio	Crea un directorio en el host remoto.
mget [archivos]	Transfiere archivos, usando wilcards desde el hosts remoto.
mput [archivos]	Envía archivos al host remoto, usando wilcards.
open nombre-host	Establece conexión con el host remoto.

prompt	Si esta "off", múltiples archivos son procesados sin pedir confirmación para cada uno.
put arch-local [arch-rem]	Envía un archivo al host remoto.
pwd	Muestra el nombre del directorio actual del host remoto.
quit	Termina sesión de FTP.
rename nombre1 nombre2	Cambia de nombre al archivo nombre1 del host remoto.
rmdir directorio	Borra directorio del host remoto.
status	Muestra los parámetros para transferencia de archivos .
user [user-id [password]]	Realiza login en el host remoto.

- **TRIVIAL FILE TRANSFER PROTOCOL (tftp):** Ofrece un subconjunto de funciones del ftp. Permite copiar archivos entre sistemas que soporten tftp; no ejecuta autenticacion de usuarios, por esto los archivos deben poderse leer y escribir por todos los usuarios . El formato del comando es:

  Para enviar archivos:

  tftp remotehost put sourcefile [targetfile]

  Para recibir archivos:

  tftp remotehost get sourcefile [targetfile]

  Para entrar a nivel de subcomandos:

  tftp [nombrehost]

Entramos al servicio, cuyo prompt es "tftp>", donde disponemos de los siguientes subcomandos:

connect [nombrehost] Realizamos conexión con el sistema remoto.

get [nombrehost:]arch-remoto .... arch-local

put arch-local .... [nombrehost:]direc-remto

quit /* Salir de tftp .

? Ayuda.

6.3.5. NFS (NETWORK FILE SYSTEM)

NFS es un servicio de red que permite a los usuarios accesar los sistemas de archivos y directorios de otros equipos en la red. Los Hosts pueden ser de diferente marca y sistema operacional. Estas diferencias son transparentes para los usuarios.

Se debe tener en cuenta los siguientes conceptos:

Servidor: El hosts que permite el acceso a sus sistemas de archivos o directorios locales (este debe exportar sus recursos).

Cliente : El hosts que solicita el acceso a sistemas de archivos o directorios de otras máquinas (este debe montar los recursos exportados por otras máquinas).

6.3.5.1. COMANDO EXPORTFS.

Permite poner a disposición de otros host, sus sistemas de archivos o directorios locales.

Este comando arranca leyendo el archivo /**etc**/**exports**. Si el archivo exports es modificado, se debe volver a correr comando exportfs para actualizar.

Opciones Básicas.

-a Exporta todos los recursos listados en /etc/exports.

-u Termina el export del recurso escrito a continuación.

-v Muestra mensajes de salida durante el proceso.

Archivo /etc/exports. El formato de este archivo es:

pathname opciones

Donde :

pathname: Es el sistema de archivos o directorio que se va a colocar a disposición de los otros host.

opciones: ro Solo lectura
rw Lectura y escritura
access Da accesos para una determinada lista de clientes solamente.

Ejm:

```
/   -ro
/usr/demos  -ro,access=cliente1:cliente2:cliente3
/usr/catman
```

6.3.5.2. COMANDO MOUNT Y UMOUNT.

Después que el servidor a colocado a disposición sus recursos, desde la máquina cliente se deben montar estos. Después de utilizados, se deben desmontar; para lo cual existen los comandos mount y umount respectivamente. Las opciones básicas de estos son:

-t Define el tipo a ser montado (Debe ser **nfs**).
-a Intenta montar todas las entradas listadas en el archivo /etc/fstab, o desmontar las encontradas en el archivo /etc/mtab.

-h Intenta montar o desmontar las entradas, para un host en particular.
-o Lee las opciones desde la línea de comando y no desde el archivo.

6.3.5.3. ARCHIVO /ETC/FSTAB.

El formato de este es:

file-sytem mount-point type options frec pass

Donde:

file-system: Directorio remoto del servidor a ser montado.

mount-point: Directorio local donde se montará el recurso.

type: Debe ser nfs.

options: Pueden ser:

ro	Solo lectura
rw	Solo escritura
hard	En caso de falla, el cliente espera a que el servidor responda.
soft	En caso de falla, el cliente hace determinado número de intentos.
bg	Permite el mount en backgroun.
fg	Permite el mount en foreground
noauto	Al hacer mount, ignora esta entrada.

frec: Debe ser 0 , para este caso.

pass: Debe ser 0 , para este caso.

Ejm:

hgt:/usr/people /usr/usuarios nfs ro,hard,bg 0 0

6.3.5.4. ACTIVANDO EL SERVIDOR.

Se asume que el software ya esta instalado. Se debe tener en cuenta los siguientes pasos:

- Verificar que los daemons esten corriendo (nfsd, biod).

-Verificar el archivo /etc/exports

-Exportar los recursos:

```
# exportfs –av
```

-Verificar:

```
# exportfs
```

6.3.5.5. ACTIVANDO EL CLIENTE .

Se asume que el software ya esta instalado. Se deben tener en cuenta los siguientes pasos:

-Editar y organizar el archivo /etc/fstab

-Crear el mount-point:

```
#mkdir /usr/usuarios
```

-Montar el recurso, bien sea manualmente, o procesando el /etc/fstab.

Ejm:

```
# mount -a

# mount hgt:/usr/people  /usr/usuarios
```

7. ADMINISTRACION IRIX 6.5 SILICON GRAPHICS

7.1. INTRODUCCIÓN AL TOOLCHEST

Un vez se tiene disponible el ambiente gráfico de Iris, podemos observar en el escritorio el toolchest que nos permite acceder una serie de aplicaciones.

7.2. MANEJO DE SESIONES

Cada sesión

7.3. MANIPULACIÓN GRAFICA DE ARCHIVOS, DIRECTORIOS, PERMISOS

En la teoría general de unix, tratamos la manipulación por comandos de los sistemas de archivos. Aquí veremos como hacer esto, con la interfaz gráfica de Iris.

7.4. PROCESO DE INSTALACIÓN DE IRIS

Debemos tener los medios magnéticos respectivos. Para el caso de Iris 6.5, son un juego de 3 CDs, etiquetados

7.5. PROCESO DE BOOT DEL SISTEMA

Antes de que se tenga a disposición el Sistema Operacional IRIX, se desarrollan una serie de tareas que tienen que ver con el proceso de arranque de la máquina y del sistema .

7.5.1. DISCOS Y PARTICIONES

El disco del sistema contiene archivos y programas necesarios para bootear y cargar el sistema operativo . Este disco es divido en secciones llamadas particiones , las cuales detallaremos más adelante , pero que mencionaremos a continuación . Básicamente se necesitan 3 particiones (los números asignados , por defecto están entre "()"):

- root (0) : Contiene el sistema de archivos del / . En este están los archivos y directorios esenciales para cargar unix . También puede contener el subdirectorio /usr , donde reside el resto del sistema operativo , si es que este no está montado en una partición separada. Es conveniente tener esta última opción.
- volume header (8): Contiene el sash (shell stand alone) y particionamiento del disco .
- swap (1): Destinada para paginación .
- usr (6) : es opcional , pues puede ser un subdirectorio de / , como se mencionó anteriormente .

En la sección de manejo de discos , ampliaré la información al respecto .

7.5.2. AMBIENTES DE LA MAQUINA

Las máquinas Silicon Graphics , tienen 4 ambientes básicos de trabajo :

MODO MANTENIMIENTO.

Cuando el sistema arranca , realiza una serie de diagnósticos, y presenta la opción de entrar a un menú de cinco opciones , para realizar una serie de tareas que facilitan las labores de instalación , recuperación , diagnósticos y cargue del sistema . Con la tecla escape (Esc) , presionada instantes después del arranque , se podrá acceder este menú o pantalla gráfica , según sea el caso (servidor con consola texto , o estación con monitor gráfico) .

Las opciones disponibles a este nivel son :

```
Running power-on diagnostics...




                          Starting up the system...

               To perform system maintenance instead, press
<Esc>


System Maintenance Menu

1) Start System
2) Install System Software
3) Run Diagnostics
4) Recover System
5) Enter Command Monitor

Option? 5
Command Monitor.  Type "exit" to return to the menu.
```

- Star system . Permite continuar con el cargue en modo multiusario del sistema operativo .
- Install System . Permite entrar a la utilidad de instalación de productos o bootear de CDROM para efectual alguna tarea de mantenimiento que así lo requiera .
- Diagnostic . Permite correr una serie de diagnósticos a todos los elementos de hardware de la máquina , tales como : tarjeta gráfica , memoria , bus scsi , etc .
- Recovery . Permite recuperar el sistema desde un backup , en caso de algún daño en el disco duro del sistema operativo .
- Command Monitor . Entra al prompt monitor , que permite realizar cambios a variables para el cargue del sistema y definición de algunos parametros de arranque . Desde aquí se puede correr la utilidad hinv para ver el inventario de hardware de la máquina y observar los dispositivos que esta

detectando la máquina en el momento del arranque. También se puede accesar un shell standalone (sash) , que permite hacer tareas más especiales , tales como bootear desde CD alguna utilidad unix, listar el contenido de una partición , copiar archivos, listar un archivo .

PROM MONITOR

Es el nivel más bajo de operación . Permite interactua con la CPU, inicializar hardware, accesar variables y comandos para cambiar la secuencia de boot , o dispositivos desde se realiza este , bootear en modo monousuario. Se llega a él por la opción (5) del menu anterior .

Los comandos básicos a este nivel son :

auto	Bootea automáticamente desde el disco que esta especificado en las variables .
sash	Carga el standalone shell
printenv	Muestra el valor de las varaibles , o de una de ellas en particular si es especificada a continuación .
setenv variable valor	Cambia el valor de una variable
single	Sube el sistema en modo monousuario o single user
unsetenv var	Elimina la variable
exit	Vuelve al menú de inicio
hinv	Presenta el inventario de dispositivos de hardware detectados por la máquina en el momento del arranque .
passwd	Asigna un password para entrar al modo mantenimiento
resetpw	Borra el password definido .
boot –f device	Carga desde un dispositivo en particular . El dispositivo de especifica por controlador , unidad y superficie , antepuesto de la palabra dksc (para dispositivos scsi) , o tpsc para cintas . Para CD se debe colocar la palabra cdrom .

	ejm : boot -f cdrom(0,5,8)sashARCS boot dksc(0,1,6)/unix.hgt
init	reinicializa el promp monitor y la máquina , para detectar dispositivos que no estaban prendidos.

Estando en este nivel se puede obtener el help :

```
>> help
Commands:
 autoboot:        auto
 boot:            boot [-f FILE] [-n] [ARGS]
 date:            date [mmddhhmm[ccyy|yy][.ss]]
 exit:            exit
 help:            help or ? [COMMAND]
 initialize:      init
 inventory:       hinv [-v] [-t [-p]]
 list files:      ls DEVICE
 passwd:          passwd
 power off machine:      off
 printenv:        printenv [ENV_VAR_LIST]
 resetenv:        resetenv
 resetpw:         resetpw
 setenv:          setenv ENV_VAR STRING
 single user:     single
 unsetenv:        unsetenv ENV_VAR
 version:         version
```

Algunas variable manejadas a este nivel son :

```
>> printenv

AutoLoad=Yes
TimeZone=EST5EDT
console=d
diskless=0
dbaud=9600
volume=80
sgilogo=y
```

```
autopower=y
netaddr=192.1.1.7
eaddr=08:00:69:08:0c:05
ConsoleOut=serial(0)
ConsoleIn=serial(0)
cpufreq=200
SystemPartition=scsi(0)disk(1)rdisk(0)partition(8)
OSLoadPartition=scsi(0)disk(1)rdisk(0)partition(0)
OSLoadFilename=/unix
OSLoader=sash
```

La utilizaciónde algunas de ellas son :

AutoLoad	Especifica si la máquina booteará automáticamente , o quedará en modo mantenimiento en el momento del arranque , para hacerlo manualmente.
TimeZone	Especifica la zona horaria , para efectos de corrimientos horarios .
console	Define si la consola es gráfica (g) como los monitores de las estaciones , o texto (d) en el caso de las pantallas normales .
dbaud	Especifica la velocidad a la cual trabajará la consola .
netaddr	Define la dirección IP de la máquina
SystemPartition	Especifica la ubicación de la partición con el sash , para bootear.
OSLoadPartition	Especifica la partición con el root a cargar
OSLoadFilename	Especifica el nombre del kernel a cargar

SASH .

Es un ambiente más inteligente que el anterior , llamado standalone shell . Usado como el programa cargador del unix . Reside en el disco , como un archivo del volumen header del disco de boot . Es cargado con el comando sash del prompt monitor o desde CD como se mostró en un ejemplo anterior.

Permite bootear programas como :

- FX : Format eXercies , para particionar , formatear, adicionar badspot, correr test a discos , etc .
- IDE : Para diagnósticos de hardware
- UNIX : local o de otra máquina en la red .

También permite algunas operaciones con directorios y archivos en disco .

Los comandos disponibles a ese nivel y la forma de cargarlo :

```
>> sash
115360+19584+3136+334528+42744d+4248+6368 entry: 0x8ffa8850
Standalone Shell SGI Version 6.2 ARCS   Mar  9, 1996 (32 Bit)

sash: ?
Commands:
 boot:           boot [-f FILE] [-n] [ARGS]
 cat:            cat FILE_LIST
 copy:           cp [-b BLOCKSIZE] [-c BCOUNT] SRC_FILE
DST_FILE
 go:             go [INITIAL_PC]
 help:           help [COMMAND]
 help:           ? [COMMAND]
 inventory:      hinv [-v] [-t [-p]]
 install:        install
 ls:             ls DIRECTORY|DEVICE
 printenv:       printenv [ENV_VAR_LIST]
 setenv:         setenv ENV_VAR STRING
 unsetenv:       unsetenv ENV_VAR
 version:        version
```

Algunos ejemplos :

```
sash: cat dksc(0,1,0)/etc/group

sys::0:root,bin,sys,adm
root::0:root
daemon::1:root,daemon
bin::2:root,bin,daemon
adm::3:root,adm,daemon
mail::4:root
uucp::5:uucp
```

```
rje::8:
lp:*:9:
nuucp::10:nuucp
user::20:
other::995:
demos:*:997:
guest:*:998:
nobody:*:60001:

sash: ls dksc(0,1,0)/usr

dksc(0,1,0)/usr:
bin  var  lib  tmp
sash :
```

- Para cargar el FX en modo experto, desde CD (en la posición 5 del scsi) :

```
sash : boot -f dksc(0,5,7)/stand/fx.ARCS --x
```

- Para bootear UNIX desde otra máquina en la red, del server prueba, del archivo /usr/local/boot/unix . Para ello el programa bootp debe correr en el servidor . Este a su vez debe arrancar el tftpd (trivial file transfer protocol daemon) :

```
sash : boot -f bootp()prueba :/usr/local/boot/unix
```

UNIX.

Es el corazón del sistema operativo , frecuentemente referido como el kernel. Esta en el disco, en la partición (0) o del root. Al ser booteado, es cargado a memoria del sistema.

Unix controla el acceso a los dispositivos de hardware, crea programas que permiten el multiprocesamiento y el ambiente multiusuario de IRIX . Estos programas permiten limitar los recursos (cpu, memora, disco) , para compartirlos productivamente entre muchos programas y usuarios .

Entre los programas que arranca , están :

- sched. Llamado también scheduler, maneja el time slicing (timepo compartido) , decide quién y cuanto tiempo , tiene control sobre la cpu en un instante dado .
- vhand. Virtual memory handler, maneja el swaping.
- bdflush . Buffer to disk flush , buffer caché . Todos los I/O de disco son temporalemente almacenados en memoria antes de ir al disco, para reducir excesivo I/O . Este programa realiza este movimiento .
- init . Establece los niveles de unix y permite el cambio entre ellos . Crea y elimina una serie de procesos .

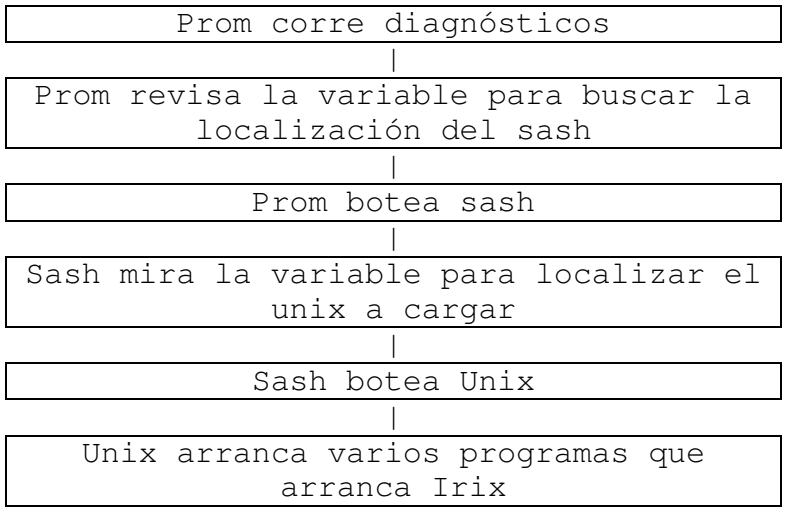
Este proceso de cargue de unix , se empieza desde el modo mantenimiento , con la opción (1) Start System , o con el comando auto desde el prompt monitor . El texto desplegado en el arranque puede variar , dependiendo de : la cantidad de productos de software configurados , el hardware , la versión de sistema operativo. Un arranque básico , puede ser :

```
Option? 1


                          Starting up the system...

IRIX Release 6.2 IP22 Version 03131015 System V
Copyright 1987-1996 Silicon Graphics, Inc.
All Rights Reserved.

WD95A SCSI controller 4 - differential external, rev 0, min
xfer period 100ns
WD95A SCSI controller 5 - differential external, rev 0, min
xfer period 100ns
The system is coming up.




www.hgt.com login:
```

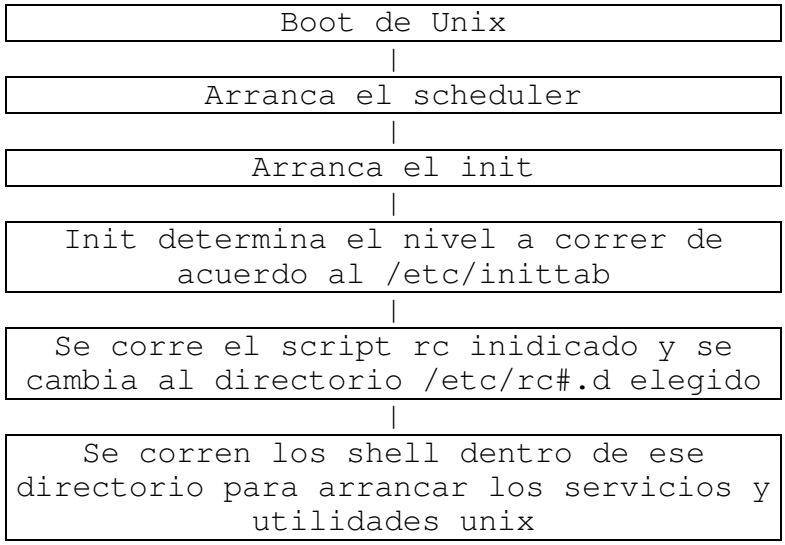
A este nivel se tienen todos los servicios disponibles y se debe proceder a dar una cuenta (login) para trabajar .

7.5.3. DIAGRAMA GENERAL DEL PROCESO DE BOOT

Para llegar al cargue de unix se realizan los siguientes pasos :



Una vez unix sea booteado , se ejecutan otras tareas :



7.5.4. ESTADOS O NIVELES IRIX

El Sistema Operacional Irix tiene varios estados:

0 El sistema se apagará.

1,s El sistema trabajará en modo monousuario.

2 Modo multiusuario.

3 Modo multiusuario y usuarios remotos.

6 El sistema se apagará y reiniciará automáticamente.

Para pasar de uno a otro estado, siempre acudirá a la tabla **/etc/inittab**, en donde encontrará el procedimiento a seguir para llegar a dicho estado. Desde sistema operativo , podemos cambiar de estado con el comando init , o shutdown , como se verá más adelante .

7.5.5. PROCESOS INICIALIZADOS EN EL ARRANQUE DEL SISTEMA Y ARCHIVOS INVOLUCRADOS.

Después del test de hardware, el sistema realiza los siguientes pasos:

- **boot**: Ejecuta el programa boot, que se encarga de subir a memoria principal el archivo **unix**.

- **init**: El unix se encarga de ejecutar el programa **init**, que a su vez verifica en

el archivo /etc/inittab, cual es el estado por defecto en el que debe quedar el sistema.

- **/etc/inittab**: En esta tabla de inicialización, además se encuentra el estado en el que quedarán las líneas directas del sistema.

- **/etc/rc2**: Una vez identifica el estado por defecto, ejecuta el programa que le corresponde, por ejemplo **rc2**, el cual a su vez busca el directorio que le corresponde a dicho estado.

- **/etc/rc2.d**: Por ejemplo para el estado multiusuario 2, le corresponde el directorio /etc/rc2.d en el cual se encuentran los siguientes procesos:

  - Actualización y montaje de sistemas de archivos.
  - Ejecuta el daemon **cron**.
  - Presentación de la actual configuración de memoria.
  - Hacer disponible el **uucp**.
  - Activar el scheduler para impresoras.
  - Activar consola.
  - Activar usuarios.

  En este directorio se puede incluir los procesos que el administrador del sistema requiera. La estructura del nombre es {**KS**}{**nn**}**nombre**, donde:

K o **S**Kill para matar procesos o Start para arrancar.
nn Número de dos cifras, indica el orden de ejecución.
nombre Nombre con que se identificará el proceso.

Todos estos archivos generalmente invocan procesos cuyos fuentes se encuentran en el directorio **/etc/init.d**.

7.5.6. APAGADO DEL EQUIPO

Existen varias formas de bajar el sistema. A través del comando:

shutdown [**-y** **-g**seg **-i**estado]

donde:

y: Responde automáticamente **yes** a todas las preguntas.

g: Permite definir a los cuantos segundos se bajará el sistema. (defecto 60 seg) (período de gracia).

i: Permite identificar a que estado se llevará el sistema. (por defecto **s**).

La forma más rápida y sencilla es ejecutando el comando:

init 0

El cual ejecuta el proceso /etc/rc0, que ejecuta los shell encontrados en el directorio /etc/rc0.d, que empiezan por **K**, e invocan los respectivos archivos en /etc/init.d.

Algunos de los procesos en /etc/rc0.d son:

- Chequea que el usuario este en /

- Chequea que el usuario sea root

- Asigna el período de gracia 60seg

- Advierte a todos los usuarios que el sistema se esta bajando.

- Mata todos los procesos que se están ejecutando

- Desmonta los sistemas de archivos

- Invoca el estado 0.

En el proceso de apagado del equipo se pueden tener varias situaciones para la cuales se pueden elegir ciertos comandos :

- Se desea bajar el sistema completamente e ir al prom , para lo cual se pueden utilizar cualquiera de los siguientes comandos :

```
init 0
halt
shutdown –y –g0
```

- Se desea pasar la máquina a modo monousuario :

```
init 1
init s
shutdown –g0 –is
```

- Se desea rebootear la máquina (bajar y subir el sistema)

```
init 6
reboot
```

7.6. MANTENIMIENTO DE USUARIOS

Desde el toolchest podemos llegar a la administración de usuarios, de la forma:

7.7. MANTENIMIENTO DE IMPRESORAS

Iris soporta los dos sistemas de impresión (System V y BSD). El sistema de system V viene por defecto instalado en el sistema. Para el sistema de BSD , se debe instalar el módulo : pinterbsd

7.8. ADMINISTRACIÓN DE PAQUETES DE SOFTWARE Y DE APLICACIONES

La instalación de paquetes de software se hace mediante el comando inst una vez el sistema este disponible.

Después de que el producto este instalado , podemos activarlo para que inicie con el sistema con el comando chkconfig.

7.9. ADMINISTRACIÓN DE DISCOS

En esta sección revisaremos los conceptos necesarios para identificación de dispositivos para discos, particionamiento de los mismos, manejo de volúmenes , mantenimiento y monitoreo .

7.9.1.IDENTIFICACIÓN DE LOS DISPOSITIVOS

Dependiendo del modelo del disco, se tienen diferentes parámetros como número de cilindros, cabezas, registros por pista, revoluciones por minuto, velocidad de

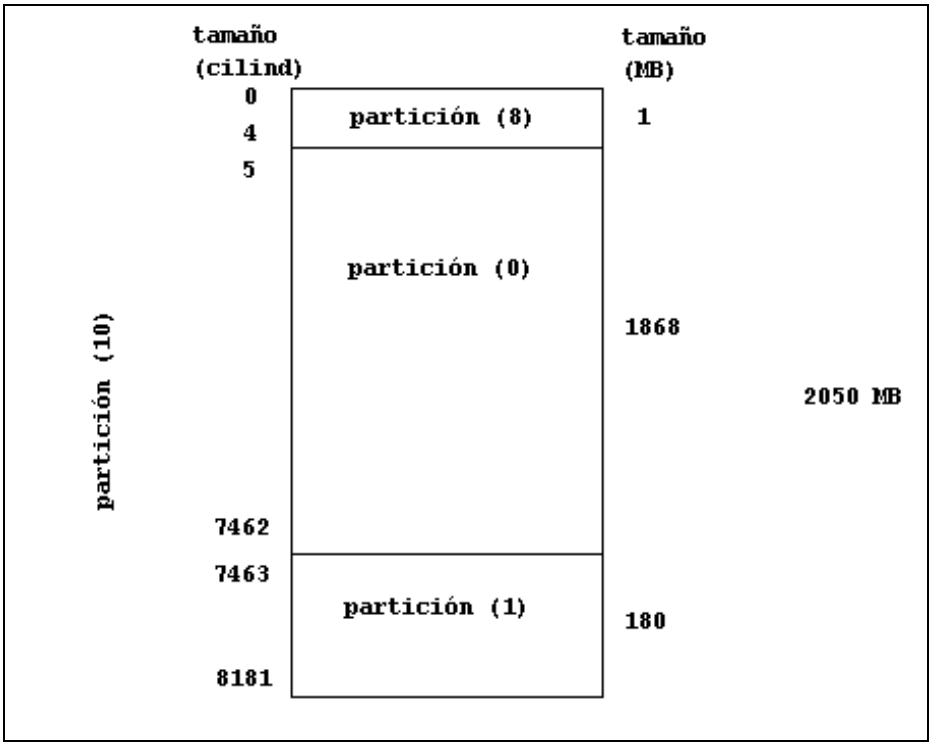
posicionamiento y otros que no tendremos en cuenta, pero que en algún momento puede influir en el rendimiento del mismo.

De esta misma forma, cada disco viene con una tabla de particiones (divisiones lógicas del disco en partes más pequeñas), que podemos modificar y utilizar en la creación de los sistemas de archivos de los usuarios.

Un ejemplo de una tabla de particiones de un disco es el siguiente:

```
part  type       cyls            blocks          Megabytes   (base+size)
  0: xfs        5 + 7458       2565 + 3826200       1 + 1868
  1: raw     7463 + 718     3828765 + 368640      1870 + 180
  8: volhdr     0 + 5           0 + 2565          0 + 1
 10: volume     0 + 8182          0 + 4197405       0 + 2050
```

Que equivaldría al siguiente diagrama de partionamiento de un disco de 2GB :



Generalmente las particiones se hacen por cilindros o por megabytes. Más adelante detallaremos la forma de modificar la tabla de particiones a nuestras necesidades.

Para la identificación del dispositivo que referencia la partición a utilizar para nuestros sistemas de archivos, necesitamos además el número del controlador al que está conectado dicho disco, así como la posición dentro de ese controlador.

Cuando mencionemos el concepto de volúmenes lógicos, introduciremos otros elementos que intervienen en esta identificación.

Por ahora tenemos que el nombre estaría compuesto de:

- Tres letras que identifican el tipo de controlador al que se esta conectado (Ejm : **dks** , para los dispositivos scsi) .
- Número que identifica la dirección del controlador (Ejm: 0, para representar la dirección cero del controlador scsi).
- La letra “d” seguida de un número que representa la posición del disco en ese controlador (Ejm : **D0,** para representar el primer disco de ese controlador).
- La letra “s” seguida de un número que representa la partición del disco que vamos a tomar (Ejm: **S0,** para representar la partición cero del disco).

Como estos dispositivos están en el directorio /dev/dsk o /dev/rdsk, el nombre del dispositivo que identifica el sector a utilizar para creación de un sistema de archivos, podría ser:

/dev/dsk/dks0d0s0

Si ese sector se va a utilizar de una forma cruda (raw), el nombre sería:

/dev/rdsk/dks0d0s0

En las versiones nuevas de Irix, tanto /dev/dsk y /dev/rdsk, son enlaces al directorio /hw.

Si desearamos ver la tabla de particionamiento de un disco, pudiésemos recurrir al comando prtvtoc direccionando la información almacenada en el volumen header del disco (partición vh), de la siguiente manera:

```
# prtvtoc /dev/rdsk/dks0d2vol

* /dev/rdsk/dks0d2vol (bootfile "/unix")
*     512 bytes/sector
* Unallocated space:
*       Start        Size
*    17780736         784
*
Partition  Type  Fs   Start: sec    Size: sec   Mount Directory
 0         xfs  yes       4096      3072000   /disco2
 1         raw          3076096       655360
 6         raw          3731456      3512320
 7         raw          7243776      3512320
 8       volhdr               0         4096
10       volume               0     17781520
11         raw          10756096      3512320
12         raw          14268416      3512320
#
```

7.9.2. ADICIONANDO UN NUEVO DISCO

Cuando se adquiere un nuevo disco , para crear nuevos sistemas de archivos , se deben tener en cuenta los siguientes pasos :

- Instalación física del disco. Se debe elegir el controlador que este menos cargado. En nuestro ejemplo, se instalará un nuevo disco en la dirección siete (7) del controlador scsi 1.

- Después de rebootear, verificar que el sistema detecto el nuevo disco, mediante el comando hinv. Por ejemplo:

```
# hinv -c disk

 Disk drive: unit 7 on SCSI controller 1
 Disk drive: unit 1 on SCSI controller 0
```

```
Integral SCSI controller 1: Version ADAPTEC 7880
Integral SCSI controller 0: Version ADAPTEC 7880
```

- Particionar el disco de acuerdo a nuestras necesidades. Para ello se utiliza el comando fx. Este comando sin opciones, invoca un modo de trabajo básico y sencillo. Si se invoca con la opción -x , llama al modo experto del mismo , que nos da más flexibilidad en el trabajo y nos permite más facilidades de trabajo, pero requiere más dominio de los conceptos asociados . En nuestro ejemplo trabajaremos con este modo, para crear cuatro (4) particiones : de 5011 MB, 1389 Mb, 1389 Mb y 892 Mb ; en el transcurso de la explicación veremos algunas tareas que se pueden realizar con el fx .

```
# fx -x
fx version 6.3, Nov 26, 1996
fx: "device-name" = (dksc)
fx: ctlr# = (0) 1
fx: drive# = (1) 7
fx: lun# = (0)
...opening dksc(1,7,0)
...controller test...OK
Scsi drive type == SEAGATE ST39173N       5958

----- please choose one (? for help, .. to quit this menu)-----
[exi]t          [d]ebug/          [l]abel/          [a]uto
[b]adblock/     [exe]rcise/       [r]epartition/    [f]ormat
```

En el primer nivel del comando fx , podemos observar las opciones para salir del comando (exit), entrar en modo depuración (debug) , observar caracteristicas del disco y grabar las modificaciones a la tabla de particiones (label), correr formateo físico del disco de una forma automática y guiada (auto) , listar y adicionar bloques defectuosos del disco (badblock) , correr test al disco (exercise) , reparticionar el disco (repartition) y formateo manual del disco (format) . No sobra mencionar que todos los discos vienen preformateados y sólo en casos de fallas continuas se debe recurrir al formateo.

Ejemplo de opción badblock:

```
fx> b
```

```
----- please choose one (? for help, .. to quit this menu)-----
[a]ddbb        [s]howbb
fx/badblock> s

total 1026 in complete defect list
Format ==  (cylinder/head/sector) (physical, not logical)
(27/5/28)      (1492/2/210)   (2281/1/149)   (3741/3/79)    (5940/0/103)
(40/3/174)     (1492/2/211)   (2282/1/86)    (3741/7/197)   (5941/0/55)
(49/6/162)     (1493/2/145)   (2283/1/23)    (3742/3/22)    (5942/0/7)
(52/9/139)     (1494/2/79)    (2284/1/226)   (3743/3/207)   (5943/0/162)
(88/5/153)     (1495/2/13)    (2285/1/163)   (3744/3/150)   (5944/0/114)
(104/1/49)     (1496/2/226)   (2286/1/100)   (3745/3/92)    (5945/0/66)
(124/1/69)     (1497/2/160)   (2287/1/37)    (3746/3/35)    (5946/0/18)
.....................................................................
SE SUPRIMIO LA SALIDA COMPLETA POR EFECTOS DE ESPACIO
.....................................................................
(1487/2/262)   (2123/5/72)    (3735/3/182)   (5694/1/106)   (7458/0/24)
(1488/2/196)   (2124/5/6)     (3736/3/124)   (5732/2/27)    (7462/5/87)
(1489/2/130)   (2125/5/251)   (3737/3/67)    (5745/5/180)
(1490/2/64)    (2215/4/194)   (3738/3/9)     (5780/7/36)
(1491/0/170)   (2279/1/9)     (3739/3/195)   (5882/8/117)
(1491/2/277)   (2280/1/212)   (3740/3/137)   (5934/7/171)
total 1026 in complete defect list

----- please choose one (? for help, .. to quit this menu)-----
[a]ddbb        [s]howbb
fx/badblock> ..

----- please choose one (? for help, .. to quit this menu)-----
[exi]t          [d]ebug/          [l]abel/          [a]uto
[b]adblock/       [exe]rcise/       [r]epartition/     [f]ormat
```

Desde la opción de exercise, podemos correr diferentes test al disco, tales como: sequenciales, escogencia de bloques aleatoriamente, sólo lectura, lectura-escritura, de una o varias pasadas, etc .

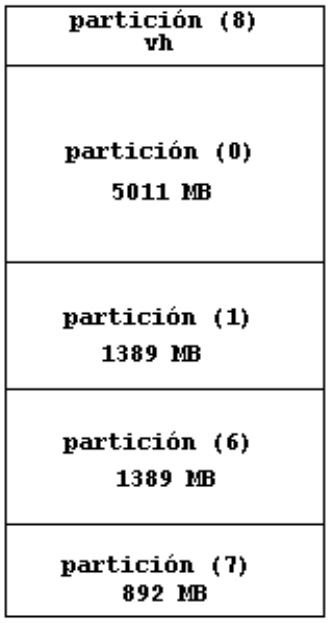
Ejemplo de opciones alcanzables desde ahí:

```
fx> exe
```

```
----- please choose one (? for help, .. to quit this menu)-----
[b]utterfly          [seq]uential          [set]testpat
[e]rrlog             [st]op_on_error       [sh]owtestpat
[r]andom             [m]iscompares         [c]omplete
fx/exercise> ..
```

Para crear o modificar particiones, recurrimos a la opción repartition. Al entrar nos muestra unas opciones con esquemas de particionamiento predefinidos, tales como: rootdrive que trae una partición (0) grande y partición (1) pequeña, o usrrootdrive que trae particiones (0) y (1) pequeñas y la partición (6) grande. Para nuestro caso, escogeremos el modo experto que nos permite más flexibilidad en la escogencia de los tamaños. Para cambiar o definir el tamaño de una partición, se puede trabajar con Mbytes o con cilindros. Debemos conocer donde inicia la partición y especificar el tamaño de la misma.

Como lo mencionamos anteriormente, deseamos llegar al siguiente esquema de particionamiento del disco de 9gb:



Para ello entramos por la opción de repartition, y luego escogemos el método de experto para mayor flexibilidad en la distribución de las particiones.

```
----- please choose one (? for help, .. to quit this menu)-----
[exi]t            [d]ebug/          [l]abel/          [a]uto
[b]adblock/       [exe]rcise/       [r]epartition/    [f]ormat
fx> r

----- partitions-----
part  type       cyls            blocks          Megabytes   (base+size)
  0: xfs        2 + 7391       4740 + 17516670       2 + 8553
  1: raw     7393 + 110    17521410 + 260700      8555 + 127
  8: volhdr     0 + 2            0 + 4740          0 + 2
 10: volume     0 + 7503         0 + 17782110      0 + 8683

capacity is 17783240 blocks

----- please choose one (? for help, .. to quit this menu)-----
[ro]otdrive          [o]ptiondrive        [e]xpert
[u]srrootdrive       [re]size
```

Aquí tenemos un bosquejo de la tabla de particiones que trae por defecto el disco y con unos cálculos sencillos podemos encontrar el equivalente entre cilindros y MB, para poder definir los tamaños de las particiones. Como ya se mencionó, lo más importante es que las particiones que se van a utilizar no se sobrepongan, es decir que la secuencia de cilindros de inicio y de cantidad de cada una de ellas, no interfiera con el número de otra partición . Pueden existir particiones en la tabla que se sobrepeongan, pero nunca se puede hacer creaciónde sistemas de archivos, o utilización de estas como raw devices, al mismo tiempo. Para lograr esto se debe tener control sobre el cilindro en donde inicia una partición y la cantidad de ellos requeridos para cumplir con la capacidad en MB, de la misma . Es importante también definir el tipo de partición que se va a definir, por ejemplo: **Raw,** para raw devices en base de datos, o para área de swap; **xfs,** para sistema de archivos en versiones de Irix superiores a la 5.3.

Se puede escoger el esquema a trabajar para particionar: por cilindros o por Megabytes (este último es más aconsejable). Los datos que se piden al particionar son :

Número de la partición a modificar de tamaño o a crear (0,1,2,3,4,5,6,7), no tomar el número 8 , ya que es el volumen header.
Tipo del sistema de archivo: xfs para sistemas de archivos tradicionales, los cuales se van a montr en un directorio; raw para dispositivos crudos para bases de datos o áreas de swap.
Cilindro o mega base : En donde empieza la partición, bien sea medida por cilindros o por megabytes.
Tamaño : número de cilindros o megas que tiene la partición.

Se pregunta por cada número de partición disponoible hasta que se termina y muestra el esquema de particiones obtenido.

Luego se debe ir al nivel de label , para grabar las modificaciones con el comando sync.

A continuación presento el diálogo que se realiza para crear el esquema anterior:

```
fx/repartition> e

Warning: you will need to re-install all software and restore user data
from backups after changing the partition layout.  Changing partitions
will cause all data on the drive to be lost.  Be sure you have the drive
backed up if it contains any user data.  Continue? y
Enter .. when done
fx/repartition/expert: change partition = (0)
before:  type xfs       base:     2 cyls,    4740 blks,    2 Mb
                        len:  7391 cyls, 17516670 blks, 8553 Mb
fx/repartition/expert: partition type = (xfs)
fx/repartition/expert: base cyl = (2)
fx/repartition/expert: number of cyls (max 7501) = (7391) 4330
 after:  type xfs       base:     2 cyls,    4740 blks,    2 Mb
                        len:  4330 cyls, 10262100 blks, 5011 Mb
fx/repartition/expert: change partition = (1)
before:  type raw       base:  7393 cyls, 17521410 blks, 8555 Mb
                        len:   110 cyls,  260700 blks,  127 Mb
fx/repartition/expert: partition type = (raw) xfs
fx/repartition/expert: base cyl = (7393) 4332
fx/repartition/expert: number of cyls (max 3171) = (110) 1200
 after:  type xfs       base:  4332 cyls, 10266840 blks, 5013 Mb
```

```
                len:  1200 cyls, 2844000 blks, 1389 Mb
fx/repartition/expert: change partition = (6)
before:  type volhdr    base:     0 cyls,       0 blks,    0 Mb
                len:     0 cyls,       0 blks,    0 Mb
fx/repartition/expert: partition type = (volhdr) xfs
fx/repartition/expert: base cyl = (0) 5532
fx/repartition/expert: number of cyls (max 1971) = (0) 1200
 after:  type xfs       base:  5532 cyls, 13110840 blks, 6402 Mb
                len:  1200 cyls, 2844000 blks, 1389 Mb
fx/repartition/expert: change partition = (7)
before:  type volhdr    base:     0 cyls,       0 blks,    0 Mb
                len:     0 cyls,       0 blks,    0 Mb
fx/repartition/expert: partition type = (volhdr) xfs
fx/repartition/expert: base cyl = (0) 6732
fx/repartition/expert: number of cyls (max 771) = (0) 771
 after:  type xfs       base:  6732 cyls, 15954840 blks, 7790 Mb
                len:   771 cyls, 1827270 blks,  892 Mb
fx/repartition/expert: change partition = (8)
before:  type volhdr    base:     0 cyls,       0 blks,    0 Mb
                len:     2 cyls,    4740 blks,    2 Mb
fx/repartition/expert: partition type = (volhdr)
fx/repartition/expert: base cyl = (0)
fx/repartition/expert: number of cyls (max 7503) = (2)
 after:  type volhdr    base:     0 cyls,       0 blks,    0 Mb
                len:     2 cyls,    4740 blks,    2 Mb

----- partitions-----
part  type       cyls            blocks          Megabytes   (base+size)
  0: xfs        2 + 4330       4740 + 10262100       2 + 5011
  1: xfs     4332 + 1200    10266840 + 2844000    5013 + 1389
  6: xfs     5532 + 1200    13110840 + 2844000    6402 + 1389
  7: xfs     6732 + 771     15954840 + 1827270    7790 + 892
  8: volhdr     0 + 2            0 + 4740         0 + 2
 10: volume     0 + 7503         0 + 17782110       0 + 8683

capacity is 17783240 blocks
```

Las particiones 8 (volume header) y 10 (volume) , no se deben utilizar , ya que estas son reservadas .

Una vez tengamos el esquema de particionamiento deseado , se debe grabar este , con la opción **sync** , dentro del submenú de **label** , así :

```
----- please choose one (? for help, .. to quit this menu)-----
[ro]otdrive          [o]ptiondrive        [e]xpert
[u]srrootdrive       [re]size
fx/repartition> ..


----- please choose one (? for help, .. to quit this menu)-----
[exi]t           [d]ebug/         [l]abel/         [a]uto
[b]adblock/      [exe]rcise/      [r]epartition/   [f]ormat
fx> l

----- please choose one (? for help, .. to quit this menu)-----
[sh]ow/      [sy]nc       [se]t/       [c]reate/
fx/label> sy

writing label info to dksc(1,7,0)

----- please choose one (? for help, .. to quit this menu)-----
[sh]ow/      [sy]nc       [se]t/       [c]reate/
fx/label> ..

----- please choose one (? for help, .. to quit this menu)-----
[exi]t           [d]ebug/         [l]abel/         [a]uto
[b]adblock/      [exe]rcise/      [r]epartition/   [f]ormat
fx> exit
```

- Creación de los sistemas de archivos, en las particiones creadas y que así lo requieran. Si la partición se va a utilizar de una forma cruda (raw), no es necesario realizar los pasos siguientes.

```
# mkfs_xfs /dev/rdsk/dks1d7s0

meta-data=/dev/rdsk/dks1d7s0     isize=256    agcount=8, agsize=160346 blks
data     =                       bsize=4096   blocks=1282762, imaxpct=25
```

```
log      =internal log          bsize=4096   blocks=1000
realtime =none                  extsz=65536  blocks=0, rtextents=0

# mkfs_xfs /dev/rdsk/dks1d7s1

meta-data=/dev/rdsk/dks1d7s1     isize=256    agcount=8, agsize=44438 blks
data     =                       bsize=4096   blocks=355500, imaxpct=25
log      =internal log          bsize=4096   blocks=1000
realtime =none                  extsz=65536  blocks=0, rtextents=0

# mkfs_xfs /dev/rdsk/dks1d7s6

meta-data=/dev/rdsk/dks1d7s6     isize=256    agcount=8, agsize=44438 blks
data     =                       bsize=4096   blocks=355500, imaxpct=25
log      =internal log          bsize=4096   blocks=1000
realtime =none                  extsz=65536  blocks=0, rtextents=0

# mkfs_xfs /dev/rdsk/dks1d7s7

meta-data=/dev/rdsk/dks1d7s7     isize=256    agcount=8, agsize=28551 blks
data     =                       bsize=4096   blocks=228408, imaxpct=25
log      =internal log          bsize=4096   blocks=1000
realtime =none                  extsz=65536  blocks=0, rtextents=0
```

El comando mkfs o mkfs _xfs, realiza la creación del sistema de archivos en cada partición de disco a utilizar. En las versiones nuevas de Irix, el tamaño del bloque para manejo del sistema de archivos es definido en 4KB (4096 bytes), por defecto. En versiones de Irix anteriores, era necesario especificarlo en el momento de la creación, pues por defecto lo definia como 512 byte. En esos casos el comando sería (para Irix 5.3 with xfs):

```
# mkfs -t xfs -d name=dks0d2s7 -b size=4k -l internal,size=2000b
```

- Creación de los puntos de montaje de los sistemas de archivos. Los nombres de los directorios son de libre elección y sólo deben cumplir con las normas de nombres de archivos y directorios en la respectiva versión de Irix.

```
# cd /
```

```
# mkdir videos job web email
```

- Montar los sistemas de archivos, para el uso de los usuarios.

```
# mount /dev/dsk/dks1d7s0 /videos

# mount /dev/dsk/dks1d7s1 /job

# mount /dev/dsk/dks1d7s6 /web

# mount /dev/dsk/dks1d7s7 /email
```

- Verificar que los sistemas de archivos, son vistos por el sistema.

```
df -k

Filesystem            Type  kbytes    use     avail  %use Mounted on
/dev/root             xfs  1908820  1765884   142936  93  /
/dev/dsk/dks1d7s0      xfs  5127048      144  5126904   1  /videos
/dev/dsk/dks1d7s1      xfs  1418000      144  1417856   1  /job
/dev/dsk/dks1d7s6      xfs  1418000      144  1417856   1  /web
/dev/dsk/dks1d7s7      xfs   909632      144   909488   1  /email
```

- Como este montaje se pierde al rebootear la máquina, se debe incluir en un archivo, para que el montaje sea automático. Este archivo es el **/ect/fstab**. Se debe editar el archivo e incluir las líneas que hacen referencia a los sistemas de archivos nuevos. En esta línea se especifica: nombre-dispositivo-a-montar punto-de-montaje tipo-de-sistema-de-archivos permisos,nombre-raw-device número-de-secuencia-montaje .

```
# cd /etc
# cat fstab

/dev/root /  xfs  rw,raw=/dev/rroot 0 0
```

Este es el archivo original de una máquina, donde sólo se esta montando el sistema de archivos del root (/). A este se le adcionan las líneas de los otros sistemas de archivos y debe quedar así:

```
# cat fstab

/dev/root /  xfs  rw,raw=/dev/rroot 0 0
/dev/dsk/dks1d7s0 /videos xfs rw,raw=/dev/rdsk/dks1d7s0 0 0
/dev/dsk/dks1d7s1 /job    xfs rw,raw=/dev/rdsk/dks1d7s1 0 0
/dev/dsk/dks1d7s6 /web    xfs rw,raw=/dev/rdsk/dks1d7s6 0 0
/dev/dsk/dks1d7s7 /email  xfs rw,raw=/dev/rdsk/dks1d7s7 0 0
```

Este archivo, también es leído cuando se invocan las órdenes :

```
# umount -a

# mount -a
```

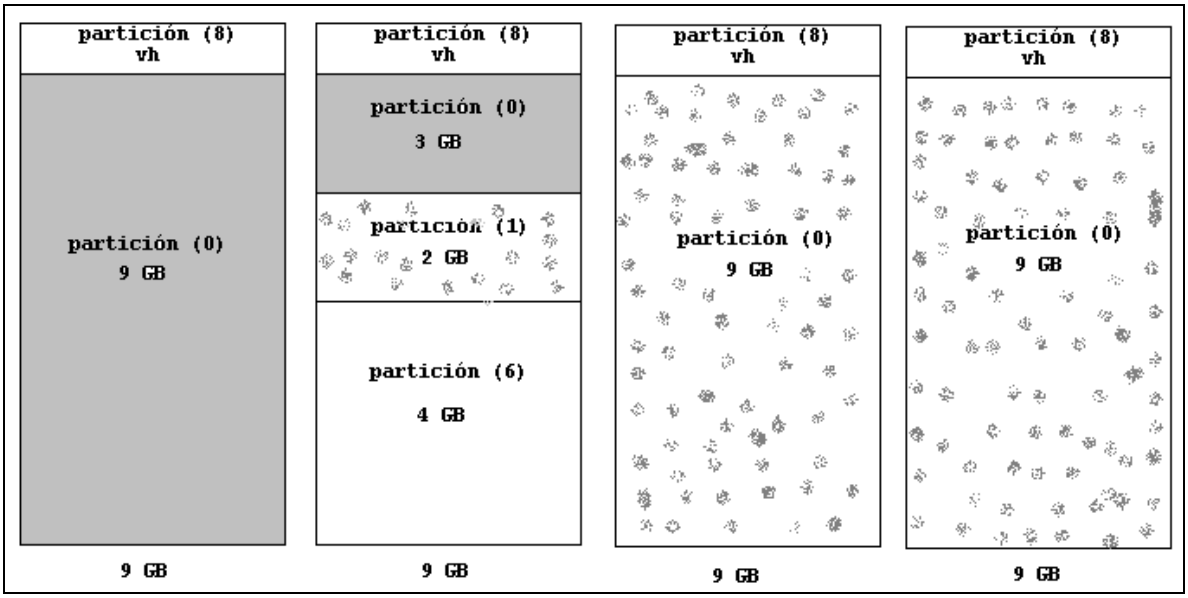
Las cuales desmontan y montan todos los sistemas de archivos a excepción del root, que figuren en él .

7.9.3.MANEJO DE VOLUMENES LÓGICOS.

Ya se trato el concepto de partición, como un segmento más pequeño de un disco físico. El concepto de vólumen lógico hace a referencia a una partición que no sólo está en un disco físico, sino que puede abarcar varios discos, logrando así tamaños superiores a los de los discos físicos. Para esbozar mejor esto, pensemos en que tenemos cuatro discos de 9 GB, y deseamos unas particiones de 12 y 20 GB, para almacenar videos e imágenes de satélite respectivamente. Sin el concepto de volúmenes, no se podría. En la actualidad, para resolver esto, podríamos tomar todo un disco de 9GB y adicionarle 3 GB de otro, para lograr la partición de 12 GB . Para la de 20 GB, podemos tomar 2 discos de 9GB y un pedazo de 2 GB. Este esquema, quedaría de la siguiente forma:

Así el área más oscura, nos daría la partición de 12 GB y el área con manchas la partición de 20 GB. Estas áreas, en realidad se denominarían volúmenes. En la creación del volúmen, es necesario declarar las particiones de discos involucradas.

Una vez creado este, se maneja semejante a las particiones de disco normales, es decir, como el volumen lo representa un dispositivo (archivo), se hace referencia a ese archivo para la creación del sistema de archivos y luego para el montaje del mismo.



Para ver ya los comandos involucrados en la creación y manipulación de volúmenes, tomaremos la siguiente configuración, en la cual se deben crear 3 volúmenes de 36 GB cada uno:

- Catorce (14) discos scsi de 9 GB, cada uno.
- Dos (2) de ellos en el controlador scsi (0), que atiende los discos internos a la máquina.
- Seis (6) discos en una gabinete externo (vault) conectados al controlador scsi (1).
- Los otros seis (6) discos en otro gabinete externo (vault) conectados al controlador scsi (2).

Para efectos de balanceo de carga de los discos, y controladores, se determino crear los volúmenes no tomando los 4 discos del mismo controlador, sino que tomar dos (2) de uno y los otros dos (2) del otro, como lo muestra el diagrama de la siguiente página.

Las instrucciones involucradas en la creación, definición, y montaje de los volúmenes, como sistemas de archivos son:

- Identificación de los dispositivos físicos:

```
# hinv

FPU: MIPS R10010 Floating Point Chip Revision: 0.0
CPU: MIPS R10000 Processor Chip Revision: 2.6
4 195 MHZ IP27 Processors
Main memory size: 1024 Mbytes
Instruction cache size: 32 Kbytes
Data cache size: 32 Kbytes
Secondary unified instruction/data cache size: 4 Mbytes
Integral SCSI controller 0: Version QL1040B (2 DISCOS INTERNOS Y CDROM)
  Disk drive: unit 1 on SCSI controller 0
  Disk drive: unit 2 on SCSI controller 0
  CDROM: unit 6 on SCSI controller 0
Integral SCSI controller 1: Version QL1040B      (6 DISCOS EN UN VAULT)
  Disk drive: unit 1 on SCSI controller 1
  Disk drive: unit 2 on SCSI controller 1
  Disk drive: unit 3 on SCSI controller 1
  Disk drive: unit 4 on SCSI controller 1
  Disk drive: unit 5 on SCSI controller 1
  Disk drive: unit 6 on SCSI controller 1
Integral SCSI controller 2: Version QL1040B      (6 DISCOS EN UN VAULT)
  Disk drive: unit 1 on SCSI controller 2
  Disk drive: unit 2 on SCSI controller 2
  Disk drive: unit 3 on SCSI controller 2
  Disk drive: unit 4 on SCSI controller 2
  Disk drive: unit 5 on SCSI controller 2
  Disk drive: unit 6 on SCSI controller 2
Integral SCSI controller 3: Version QL1040B      (CONTROLADORES DE LA  XIO)
Integral SCSI controller 4: Version QL1040B (rev. 2)
Integral SCSI controller 5: Version QL1040B
IOC3 serial port: tty1
IOC3 serial port: tty2
Integral Fast Ethernet: ef0, version 1
IOC3 external interrupts: 1
```

- Verificar los file system montados :

```
# df -k
Filesystem            Type  kbytes     use     avail  %use Mounted on
/dev/root             xfs  8360424   771136  7589288  10  /
/dev/dsk/dks0d2s7      xfs  8880616      176  8880440   1  /disco1
```

- Crear o definir los volúmenes con el xlv_make :

```
# xlv_make
xlv_make> vol xlv0        (Definir el volúmen xlv0 con 4 particiones)
xlv0
xlv_make> data
xlv0.data
xlv_make> plex
xlv0.data.0
xlv_make> ve dks1d1s7     (Se especifican las particiones que lo componen)
xlv0.data.0.0
xlv_make> ve dks1d2s7
xlv0.data.0.1
xlv_make> ve dks2d1s7
xlv0.data.0.2
xlv_make> ve dks2d2s7
xlv0.data.0.3
xlv_make> end             (Finalización de la declaración del volúmen xlv0)
Object specification completed
xlv_make> show            (Ver la definición )

      Completed Objects
(1)  VOL xlv0 (empty)          (node=NULL)
VE xlv0.data.0.0       [empty]
      start=0, end=17769231, (cat)grp_size=1
      /dev/dsk/dks1d1s7 (17769232 blks)
VE xlv0.data.0.1       [empty]
      start=17769232, end=35538463, (cat)grp_size=1
      /dev/dsk/dks1d2s7 (17769232 blks)
VE xlv0.data.0.2       [empty]
```

```
      start=35538464, end=53307695, (cat)grp_size=1
      /dev/dsk/dks2d1s7 (17769232 blks)
VE xlv0.data.0.3      [empty]
      start=53307696, end=71076927, (cat)grp_size=1
      /dev/dsk/dks2d2s7 (17769232 blks)

xlv_make> vol xlv1          (Definir el volúmen xlv1 con 4 particiones)
xlv1
xlv_make> data
xlv1.data
xlv_make> plex
xlv1.data.0
xlv_make> ve dks1d3s7
xlv1.data.0.0
xlv_make> ve dks1d4s7
xlv1.data.0.1
xlv_make> ve dks2d3s7
xlv1.data.0.2
xlv_make> ve dks2d4s7
xlv1.data.0.3
xlv_make> end
Object specification completed
xlv_make> vol xlv2          (Definir el volúmen xlv2 con 4 particiones)
xlv2
xlv_make> data
xlv2.data
xlv_make> plex
xlv2.data.0
xlv_make> ve dks1d5s7
xlv2.data.0.0
xlv_make> ve dks1d6s7
xlv2.data.0.1
xlv_make> ve dks2d5s7
xlv2.data.0.2
xlv_make> ve dks2d6s7
xlv2.data.0.3
xlv_make> end
Object specification completed
xlv_make> exit              (Al salir se realiza la creción de los volúmenes)
Newly created objects will be written to disk.
```

```
Is this what you want?(yes)
Invoking xlv_assemble
#
```

- En la mayoría de versiones de Irix, se pide rebootear el sistema

```
#reboot
```

- Verificar que los dispositivos asociados a los volúmenes fueron creados. la estructura o ubicación del directorio , dentro de /dev , puede variar , según la versión.

```
# cd /dev/xlv
# ls
xlv0  xlv1  xlv2

# ls -l
total 0
brw-------   1 root     sys      192,  6 Oct  8 21:45 xlv0
brw-------   1 root     sys      192,  5 Oct  8 21:45 xlv1
brw-------   1 root     sys      192,  4 Oct  8 21:45 xlv2
```

- Crear los directorios para montar los sistemas de archivos necesarios, si es el caso. Cuando se van a usar en modo “raw device”, no es necesario.

```
# cd /
# mkdir disco2 disco3 disco4
```

- Crear los sistemas de archivos en los volúmenes que asi lo requieran.

```
#mkfs /dev/xlv/xlv0

meta-data=/dev/xlv/xlv0          isize=256    agcount=34, agsize=261313 blks
data     =                       bsize=4096   blocks=8884616, imaxpct=25
log      =internal log           bsize=4096   blocks=1000
realtime =none                   extsz=65536  blocks=0, rtextents=0

# mkfs /dev/xlv/xlv1
```

```
meta-data=/dev/xlv/xlv1          isize=256    agcount=34, agsize=261313 blks
data     =                       bsize=4096   blocks=8884616, imaxpct=25
log      =internal log           bsize=4096   blocks=1000
realtime =none                   extsz=65536  blocks=0, rtextents=0

# mkfs /dev/xlv/xlv2

meta-data=/dev/xlv/xlv2          isize=256    agcount=34, agsize=261313 blks
data     =                       bsize=4096   blocks=8884616, imaxpct=25
log      =internal log           bsize=4096   blocks=1000
realtime =none                   extsz=65536  blocks=0, rtextents=0
```

- Realizar el montaje de los sistemas de archivos en los directorios respectivos.

```
# mount /dev/xlv/xlv0 /disco2
# mount /dev/xlv/xlv1 /disco3
# mount /dev/xlv/xlv2 /disco4
```

- Verificar que los sistemas de archivos fueron montados:

```
# df -k

Filesystem            Type  kbytes     use     avail  %use Mounted on
/dev/root             xfs  8360424   771136  7589288  10  /
/dev/dsk/dks0d2s7      xfs  8880616      176  8880440   1  /disco1
/dev/xlv/xlv0          xfs 35534464      560 35533904   1  /disco2
/dev/xlv/xlv1          xfs 35534464      560 35533904   1  /disco3
/dev/xlv/xlv2          xfs 35534464      560 35533904   1  /disco4
```

- Hacer la inclusión de las líneas repectivas en el archivo /**etc**/**fstab**, para que estos sistemas de archivos sean montados siempre con el arranque de la máquina.

- Se puede visualizar la definición completa de los volúmenes creados, desde el comando xlv_make:

```
# xlv_make

xlv_make> show
```

```
      Completed Objects
(1)  VOL xlv2 (complete)              (node=limonal)
VE xlv2.data.0.0       [active]
      start=0, end=17769231, (cat)grp_size=1
      /dev/dsk/dks1d5s7 (17769232 blks)
VE xlv2.data.0.1       [active]
      start=17769232, end=35538463, (cat)grp_size=1
      /dev/dsk/dks1d6s7 (17769232 blks)
VE xlv2.data.0.2       [active]
      start=35538464, end=53307695, (cat)grp_size=1
      /dev/dsk/dks2d5s7 (17769232 blks)
VE xlv2.data.0.3       [active]
      start=53307696, end=71076927, (cat)grp_size=1
      /dev/dsk/dks2d6s7 (17769232 blks)

(2)  VOL xlv0 (complete)              (node=limonal)
VE xlv0.data.0.0       [active]
      start=0, end=17769231, (cat)grp_size=1
      /dev/dsk/dks1d1s7 (17769232 blks)
VE xlv0.data.0.1       [active]
      start=17769232, end=35538463, (cat)grp_size=1
      /dev/dsk/dks1d2s7 (17769232 blks)
VE xlv0.data.0.2       [active]
      start=35538464, end=53307695, (cat)grp_size=1
      /dev/dsk/dks2d1s7 (17769232 blks)
VE xlv0.data.0.3       [active]
      start=53307696, end=71076927, (cat)grp_size=1
      /dev/dsk/dks2d2s7 (17769232 blks)

(3)  VOL xlv1 (complete)              (node=limonal)
VE xlv1.data.0.0       [active]
      start=0, end=17769231, (cat)grp_size=1
      /dev/dsk/dks1d3s7 (17769232 blks)
VE xlv1.data.0.1       [active]
      start=17769232, end=35538463, (cat)grp_size=1
      /dev/dsk/dks1d4s7 (17769232 blks)
VE xlv1.data.0.2       [active]
      start=35538464, end=53307695, (cat)grp_size=1
      /dev/dsk/dks2d3s7 (17769232 blks)
```

```
VE xlv1.data.0.3        [active]
      start=53307696, end=71076927, (cat)grp_size=1
      /dev/dsk/dks2d4s7 (17769232 blks)

xlv_make> exit
xlv_assemble not invoked
```

7.9.4.COMANDOS VARIOS ASOCIADOS AL MANTENIMIENTO Y OPERACIONES CON DISCOS Y PARTICIONES

Dentro de estos mencionaremos los comandos que permiten realizar las siguientes tareas:

- **mkfs**

Permite crear los sistemas de archivos .

Sintaxis :

```
mkfs -t efs opciones-efs  dispositivo
mkfs -t xfs opciones-xfs dispositivo
```

Permite la construcción de sistemas de archivos , escribiendo sobre el archivo de dispositivo especificado en la línea de comandos . El sistema de archivos construido, puede ser EFS o XFS. En las versiones de Irix superiores a la 6.x, se crean por defecto tipo XFS.

- **xfs_check**

Chequea las inconsistencias en los sistemas de archivos.

Sintaxis :

xfs_check –opciones xfs_special

Normalmente corre cuando se tiene alguna razón para creer que el sistema de archivos tiene problemas. El sistema de archivos a ser chequeado debe ser especificado por el nombre del disco o dispositivo del volumen.

- **xfs_repair**

Permite reparar un sistema de archivos.

Sintaxis :

xfs_repair –opciones xfs_special

Repara sistemas de archivos XFS corruptos.

- **xlv_mgr**

Administrador de los XLV logical volume y sus disk labels. Muestra y permite modificar los objetos tales como : volúmenes, plexes, elementos, disk labels, etc. , aún cuando ellos están montados y en uso.

Al invocarlo se entra a un nivel de subcomandos que permiten realizar las tareas anteriormente mencionadas.

Show : Muestra la lista de objetos XLV en el sistema y su s estructura.
Change: Cambia atributos asociados con los objetos.

- **xlv_assemble**

Inicializa los objetos de los volúmenes , desde los labels del disco. Busca por todos los discos conectados al sistema local y ensambla todos los volúmenes lógicos y genera su estructura. También crea los dispositivos necesarios asociados. Corre automáticamente con el arranque de la máquina.

- **xlv_shutdown**

Baja los volúmenes. Realiza el proceso contrario al ensamblaje, después de que los sistemas de archivos correspondientes han sido desmontados.

- **xfs_copy**

Copia el contenido el un sistema de archivos XFSa uno o más destinos. Se debe especificar el raw device. Debe ser usado para copiar sistemas de archivos desmontados o montados read-only.

Sintaxis:

xfs_copy device-file device1 device2 device3 ...

- **xfs_growfs**

Permite expandir o crecer un sistema de archivos que está actualmente montado. El espacio es adicionado sin dañar la información actual.

7.10. ADMINISTRACIÓN DE LA RED

7.11. ADMINISTRACIÓN DE SERVICIOS DE INTERNET

- Para la creación manual de estas impresoras , se debe verificar que el módulo respectivo del sistema operacional este instalado :

```
# versions |grep bsd

I  print.sw.bsdlpr      08/27/97  Berkeley 'lpr' Printer Spooler
```

- Editar el archivo /etc/printcap , que contiene las definiciones de estas impresoras e incluir las líneas respectivas para su creación . El significado de los parámetros a incluir en este archivo son :

```
    nombre-impresora| comentarios :\
    :rm=nombre-maquina-remota-que-contiene-impreora:\
    :rp=nombre-del-puerto-remoto-o-cola-de-impresion-
remota:\
    :sd=/var/spool/lpd/nombre-directorio:
```

El nombre de la impresora , es el nombre local , con e cual se desea referenciar la impresora remota. No necesariamente debe ser el nombre real o remoto de la impresora.

El nombre de la máquina remota , a donde esta conectada la impresora , debe estar incluido con su respectiva dirección IP

en el archivo /etc/hosts . En caso contrario se puede incluir directamente la dirección IP de la máquina en este campo.

En el caso de los servidores de impresión, donde sus puertos (seriales y paralelos) se identifican por un nombre de servicio , este , se debe colocar a continuación del parámetro **rp=** . Para impresoras conectadas a servidores , o PCs que pueden compartir sus impresoras con la red (bajo tcp/ip) , en ese parámetro se especifica el nombre de la cola de impresión o impresora remota.

Con el parámetro **sd=** se especifica el nombre del directorio local , donde se desea se almacene la cola de impresión . En ese directorio de deben tener ciertos permisos .

un ejemplo de archivo printcap es el siguiente :

```
# cd /etc

# cat printcap

#
# Copyright (c) 1983 Regents of the University of California.
# All rights reserved.
#
# Redistribution and use in source and binary forms are
permitted
# provided that this notice is preserved and that due credit
is given
# to the University of California at Berkeley. The name of
the University
# may not be used to endorse or promote products derived from
this
# software without specific prior written permission. This
software
# is provided ``as is'' without express or implied warranty.
#
#       @(#)etc.printcap        5.2 (Berkeley) 5/5/88
#
# DecWriter over a tty line.
#lp|ap|arpa|ucbarpa|LA-180 DecWriter III:\
```

```
#
:br#1200:fs#06320:tr=\f:of=/usr/lib/lpf:lf=/var/adm/lpd-errs:
# typical remote printer entry
#ucbvax|vax|vx|ucbvax line printer:\
#       :lp=:rm=ucbvax:sd=/var/spool/vaxlpd:lf=/var/adm/lpd-
errs:
hplce:\
     :rm=sunab18:\
     :sd=/var/spool/lpd/hplce:

p5000:\
     :rm=M_030d76:\
     :rp=d1prn:\
     :sd=/var/spool/lpd/p5000:
```

- Crear el directorio para la cola del spool y dar los permisos necesarios . Ejemplo :

```
#cd /var/spool/lpd

#mkdir p5000

# chmod 777 p5000
```

- Proceder con los comandos necesarios para operación de este sistema de spool . Son muy semejantes a los del sistema de spool anterior y se pueden invocar de dos formas : Desde la línea de comandos del **lpc** con opciones y argumentos , o dentro de un nivel de subcomandos del comando **lpc** .

Los comandos más utilizados para la manipulación de estas impresoras son :

lpc : Permite dar los subcomandos para operación del spool .

lpq : Permite revisar el estado de la cola de impresión de una impresora.

Desde el diálogo del **lpc** , se pueden dar otros subcomandos para habilitar, detener , reiniciar ,etc las impresoras (especificadas como argumento) , tales como :

```
abort          : Abortar la impresión de un trabajo
enable         : Habilitar la impresión de trabajos
disable        : Deshabilita la impresión de trabajos
help   o ?     : Presenta la ayuda
restart        : Reinicia el trabajo de una impresora y su
daemon
status         : Presenta el status de una o todas las
impresoras
exit           : Salir
quit           : Salir
start              : Arranca el servicio de una impresora y
su daemon
stop           : Detiene el servicio de una impresora y su
daemon
```

Ejemplos :

```
# lpc status

hplce:
        queuing is enabled
        printing is enabled
        1 entry in spool area
        no daemon present
p5000:
        queuing is enabled
        printing is enabled
        2 entries in spool area
        no daemon present

www 6# lpq -Phplce

Warning: no daemon present
Rank   Owner      Job  Files
Total Size
1st    root       0    /etc/group
411 bytes
```

```
www 7# lpc

lpc> ?
Commands may be abbreviated.  Commands are:

abort   enable  disable help    restart status  topq    ?
clean   exit    down    quit    start   stop    up
lpc> stop hplce
hplce:
        printing disabled

lpc> enable hplce
hplce:
        queuing enabled

lpc> disable hplce
hplce:
        queuing disabled

lpc> start hplce
hplce:
        printing enabled
        daemon started

lpc> quit
```

En todos los sistemas de computo , es indispensable tener copia de la información , en algún medio , para que sea recuperada en el momento que así se requiera . Dentro de las políticas de backups o de seguridad de la información , se puede pensar en :

- Tener una copia total de la información en un medio magnético (backup completo, podría ser quincenal o mensual) y luego sacar backup de la información que ha sido modificada , desde la copia anterior (backup incremetal, pueden ser diarios). Esto con el fin de disminuir el tiempo de backup , ya que los incrementales son mucho más cortos .

Aunque , en el proceso de recuperación , puede ser mucho más demorado .

- Sacar siempre backups completos . Estos han disminuido en tiempo gracias a la evolución de los medios magnéticos , y dependen mucho de la cantidad de información a almacenar.
- Pensar en sistemas de disco organizados en lo que se conoce como Mirror (discos espejo) , en los cuales siempre existe la información duplicada en otro disco , que remplazaría a la del disco original , en el caso que se requiera.
- Existen otras formas de organización de los discos , que permiten tener un sistema confiable , pero tanto en la anterior , como en estas , es recomendable pensar en los backup en algún medio magnetico/óptico .

Recordemos que hay diferentes tipos de medio magnético , para la realización de los backups . Existen unidades de cinta de 9 track (en su mayoría descontinuadas) , unidades de tape Exabyte o de 8 mm , unidades de tape DAT o de 4mm, arreglos de unidades con capacidades superiores .

Las unidades de almacenamiento , son encontradas en Irix , en el directorio /dev/rmt y su nombre depende del controlador y la posición en la cual se conectan . Para tener información al respecto se debe dar el comando **hinv** , como se muestra a continuación :

```
#hinv -c tape
Tape drive: unit 6 on SCSI controller 1: DAT
```

Aquí se puede observar que hay una unidad de backup DAT (4mm) , conectada al controlador scsi 1 , en la posición 6 . De esa posición depende el nombre que a ella se le asigna , siguiendo la nomenclatura :

tpsN**d**MPPP

donde :
N : Número del controlador a donde esta conectada la unidad .
M : Posición en el controlador en donde se conectó la unidad.
PPP : Puede tener los siguientes valores :

nr : El dispositivo no rebobina después de una operación
v : El dispositivo puede manejar registros de longitud variable.
C : Grabar en modo comprimido , duplicando su capacidad.

Existen más valores a tomar , pero no se detallarán . Con esta nomenclatura , el nombre de los dispositivos que se crean para un determinado medio magnético , pueden ser :

```
#pwd
/dev
#cd rmt
#ls
tps1d6       tps1d6nrnsv  tps1d6nrv    tps1d6s      tps1d6v
tps1d6nr     tps1d6nrs    tps1d6ns     tps1d6stat
tps1d6nrns   tps1d6nrsv   tps1d6nsv    tps1d6sv
```

Si deseáramos ver el estado de la unidad de cinta , se debe dar el comando :

```
mt -t /dev/rmt/tps1d6 status
        Controller: SCSI
        Device: SONY: SDT-9000        12.2
        Status: 0x200
        Drive type: DAT
        Media : Not READY
```

Para la realización de los backups , tenemos en Irix varios comandos , los cuales discutiremos a continuación :

3.1 COMANDO TAR

Es usado para almacenar y restaurar archivos y directorios. Es más poderoso y útil cuando se hace backup de archivos simples o directorios, que cuando se hace de sistemas de archivos completos. El formato del comando es:

tar **-**opciones archivo1 archivo2 archivo3 ...

donde :

opciones :**c**: Escribe en la cinta.

x: Lee de la cinta.

v: Modo verbose.

t: Lista el contenido de la cinta.

u: Escribe solo si los nombres dados no existen en la cinta o si han sido modificados desde la última vez que fueron escritos.

f: Especifica que el siguiente argumento es el nombre del dispositivo a usar.

Ejemplos:

Para sacar backup del subdirectorio pruebas, dentro de /usr/people/hector:

```
#tar -cv /usr/people/hector/pruebas

a /usr/people/hector/pruebas/hector 17 blocks
a /usr/people/hector/pruebas/hector1 61 blocks
a /usr/people/hector/pruebas/hector2 9 blocks
a /usr/people/hector/pruebas/hector3 7 blocks
a /usr/people/hector/pruebas/hector4 17 blocks
a /usr/people/hector/pruebas/hector5 10 blocks
```

Para leer el archivo hector1 del backup hecho anteriormente:

```
# tar -xv /usr/people/hector/pruebas/hector1

x /usr/people/hector/pruebas/hector1, 30763 bytes, 61
tape blocks
```

3.2 COMANDO CPIO

Permite almacenar o restaurar de cinta archivos o directorios. Es más útil y poderoso para trabajar con archivos o directorios simples que con sistemas de archivos completos.

El formato del comando para almacenar es:

cpio **–**opciones **<**lista-de-nombres **>**dispositivo o

se puede dar la opción **–O** para especificar el dispositivo en el cual se realizará la copia (no se colocaría el signo >)

El formato para restaurar es:

cpio **–**opciones **<**dispositivo o

se puede dar la opción **–I** para especificar el dispositivo del cual se restaurará la información (no se colocaría el signo <)

Este comando comúnmente se usa en conjunción con el comando **find**, ya que este le proporciona la lista de nombres que el **cpio** va a almacenar.

donde :

opciones:

o: Escribir. la salida estándar debe ser direccionada al dispositivo a usar.

c: Escribe la información del header en formato ASCII.

d: Crea directorios si se requiere.

v: Modo verbose.

B: La salida es escrita en bloques de 5.120 bytes por registro. Esta opción aumenta la velocidad de la tarea. Si es usada para escribir, se debe usar también para leer.

i: Leer. Se direcciona el dispositivo del cual leer.

t: Lista sin restaurar.

Ejemplos:

Se realiza un backup del contenido del subdirectorio pruebas (Los archivos y directorios quedan con el pathname completo), en el dispositivo /dev/rmt/tps1d6 (DAT).

```
# find /usr/people/hector/pruebas/* -print|cpio-ocvdB
>/dev/rmt/tps1d6

/usr/people/hector/pruebas/hector
/usr/people/hector/pruebas/hector1
/usr/people/hector/pruebas/hector2
/usr/people/hector/pruebas/hector3
/usr/people/hector/pruebas/hector4
/usr/people/hector/pruebas/hector5
120 blocks
```

Se podría haber usado el comando :

```
#find /usr/people/hector/pruebas/* -print|cpio-ocvdBO
/dev/rmt/tps1d6
```

Es importante tener en cuenta si la información a almacenar esta en forma relativa o absoluta . En el caso anterior estaba en forma absoluta y de igual forma se tendría que recuperar , es decir se debe recuperar en la misma ubicación de filesystem y directorio .

Se puede realizar el mismo backup , pero los pathname son pasados en forma relativa y así al recuperarla , se puede hacer en otra ubicación .

```
# cd /usr/people/hector/pruebas
# find . -print |cpio -ovdcB >/dev/rmt/tps1d6
.
hector
hector1
hector2
hector3
hector4
hector5
120 blocks
```

Para sacar un índice del backup realizado anteriormente:

```
# cpio -itvd </dev/rmt/tps1d6

40755  root         0  Feb 25 15:38:04 1993  .
100644 root      8702  Feb 25 15:38:03 1993  hector
100644 root     30763  Feb 25 15:38:03 1993  hector1
100644 root      4355  Feb 25 15:38:04 1993  hector2
100644 root      3106  Feb 25 15:38:04 1993  hector3
100644 root      8265  Feb 25 15:38:04 1993  hector4
100644 root      4724  Feb 25 15:38:04 1993  hector5
119 blocks
```

3.3. COMANDO BRU

La sintaxis es semejante a la del comando tar , pero tiene unas ventajas adicionales , tales como :

- Comprensión de archivos
- Facilidad para localizar y almacenar archivos por fecha de codificación.
- Habilidad para calcular requerimientos de espacio .
- habilidad para chequear la integridad de la información en la cinta.

- Niveles de trace o seguimiento mientras se ejecutan operaciones de backup.

Algunos ejemplos :

- Realizar copia de archivos individuales :

  bru -c archivos

- Realizar copia de archivos , especificando los nombres de estos en otro archivo .

  bru -c ´cat archivo-nombres´

- Salvar archivos por fecha de modificación

  bru -c -n 10-Nov-1998 /usr

- Listar el contenido de una cinta

  bru -tv

- Estimar el espacio requerido para una copia

  bru -e /usr

- Especificar compresión con un algoritmo LZW de 12 bit.

  bru -cZv archivos

- Extraer la información completa de una cinta

  bru -x

- Extraer archivos individuales

  bru -x archivos

3.4. COMANDOS XFSDUMP Y XFSRESTORE

En el caso de particiones completas , existen herramientas como xfsdump y xfsrestore, que permiten realizar y recuperar backup de particiones individuales , y permiten una mayor interacción con el contenido de la cinta.

En versiones anteriores , donde existían sistemas de archivos EFS , este comando era conocido como **dump** .

La sintaxis del comando es :

dump opciones dispositivo sistema-de-archivos

Donde las opciones pueden ser :

: Número que representa el nivel del backup (0-9). Cero (0) es total .
d : Especifica la densidad de la unidad (expresada en bytes per inch BPI)
s : Especifica el tamaño de la cinta en pies .
f : Especifica la unidad donde se realizará el backup . Se debe especificar el dispositivo a continuación .

Para las versiones nuevas , el comando es xfsdump y cambia su sintaxis :

xfsdump opciones sistema-de-archivos

Donde las opciones pueden ser :

-f : Especifica la unidad donde se realizará el backup . Se debe especificar el dispositivo a continuación .
-l : Especifica el nivel del backup , con un número de 0 a 9 .
-p : Especifica cada cuantos segundos se despliega información de la operación .
-v : Especifica si se muestran mensajes o no , para ver detalles en la operación.

Ejemplos : Para realizar un backup completo solamente de la partición del root (/) , con un nivel de detalle alto , mostrando mensajes por pantalla . Se pide inicialmente el

nombre que se la dará a la cinta (label) y luego chequea si la cinta contiene información , en este caso que ya contenía , pide confirmación para sobrescribirla :

```
#xfsdump -f /dev/rmt/tps1d7 -l 0 -p 30 -v trace /
xfsdump: version 2.0 - type ^C for status and control

 ============================dump         label         dialog
==============================

please enter label for this dump session (timeout in 300 sec)
 -> root
session label entered: "root"

 ------------------------------ end dialog -------------------
--------------

xfsdump: level 0 dump of pelicano.uis.edu.co:/
xfsdump: dump date: Tue Nov  3 14:56:53 1998
xfsdump: session id: 1731c081-07a3-1022-87d2-080069056a64
xfsdump: session label: "root"
xfsdump: ino map phase 1: skipping (no subtrees specified)
xfsdump: ino map phase 2: constructing initial dump list
xfsdump: ino map phase 3: skipping (no pruning necessary)
xfsdump: ino map phase 4: skipping (size estimated in phase 2)
xfsdump: ino map phase 5: skipping (only one dump stream)
xfsdump: ino map construction complete
xfsdump: estimated dump size: 38885440 bytes
xfsdump: /var/xfsdump/inventory created
xfsdump: preparing drive
xfsdump:  fixed  block  size  noncompressing  tape  drive  at
/dev/rmt/tps1d7
xfsdump:  status  at  14:57:22:  0/849  files  dumped,  0.0%
complete, 29 seconds elapsed
xfsdump: cannot determine tape block size after two tries
xfsdump: assuming is media is corrupt or contains non-xfsdump
data
xfsdump: WARNING: media contains non-xfsdump data or a corrupt
xfsdump media file header at beginning of media

 ============================media        overwrite        dialog
============================
```

```
overwrite non-xfsdump data on media in drive 0?
1: don't overwrite (timeout in 3600 sec)
2: overwrite (default)
 -> 2
media will be overwritten

 ------------------------------ end dialog -------------------
--------------

xfsdump:  status  at  14:57:52:  0/849  files  dumped,  0.0%
complete, 59 seconds elapsed

 ============================media         label         dialog
============================

please enter label for media in drive 0 (timeout in 300 sec)
 -> root
media label entered: "root"

 ------------------------------ end dialog -------------------
--------------

xfsdump: creating dump session media file 0 (media 0, file 0)
xfsdump: dumping ino map
xfsdump: dumping directories
xfsdump: dumping directory ino 128
xfsdump: dumping directory ino 131
xfsdump: dumping directory ino 135
xfsdump: dumping directory ino 136
xfsdump: dumping directory ino 137

.............. SE SUPRIMEN LINEAS , POR EFECTOS DE LONGITUD
..............
xfsdump: dumping special file ino 262334 mode 0xa1ed
xfsdump: dumping special file ino 262335 mode 0xa1ed
xfsdump: dumping special file ino 262368 mode 0xa1ed
xfsdump: dumping special file ino 262369 mode 0xa1ed
xfsdump: dumping regular file ino 262372 offset 0 to offset
1490 (size 1490)
xfsdump: dumping regular file ino 262373 offset 0 to offset
2482 (size 2482)
xfsdump: dumping regular file ino 262374 offset 0 to offset
680 (size 680)
```

```
xfsdump: dumping regular file ino 262375 offset 0 to offset
2038 (size 2038)

.............. SE SUPRIMEN LINEAS , POR EFECTOS DE LONGITUD
..............
xfsdump: status at 14:59:52: 848/849 files dumped, 69.1%
complete, 179 seconds elapsed
xfsdump: ending media file
xfsdump: media file size 39845888 bytes
xfsdump: dumping session inventory
xfsdump: beginning inventory media file
xfsdump: media file 1 (media 0, file 1)
xfsdump: ending inventory media file
xfsdump: status at 15:00:22: 849/849 files dumped, 96.3%
complete, 209 seconds elapsed
xfsdump: dumping inventory
xfsdump: inventory media file size 4194304 bytes
xfsdump: writing stream terminator
xfsdump: beginning media stream terminator
xfsdump: media file 2 (media 0, file 2)
xfsdump: ending media stream terminator
xfsdump: media stream terminator size 2097152 bytes
xfsdump: ending stream: 217 seconds elapsed
xfsdump: I/O metrics: 3 by 2MB ring; 25/33 (76%) records
streamed; 362235B/s
xfsdump: dump complete: 217 seconds elapsed
```

Para recuperar la información de la cinta se puede hacer de forma normal o interactiva . La sintaxis del comando xfsrestore es :

xfsrestore opciones directorio-destino

donde las opciones son :

-e : No sobrescriba archivos existentes en el disco
-f : Especifica el dispositivo del cual se va a extraer la información.
-i : Entra la modo de extracción interactivo . Tiene subcomandos .
-r : Extrae información

-t : Solamente lista contenido de la cinta . Si se escoje esta opción , nmo hay necesidad de especificar directorio destino .

Ejemplo : Se listará el contenido de la cinta grabada en el ejemplo anterior. Los subcomandos listados allí , aplican también si se va a extraer información de forma interactiva .

```
#xfsrestore -i -t -v silent -f /dev/rmt/tps1d7

 ========================dump        selection        dialog
============================

the following dump has been found on drive 0

hostname: pelicano.uis.edu.co
mount point: /
volume: /dev/rroot
session time: Tue Nov  3 14:56:53 1998
level: 0
session label: "root"
media label: "root"
file system id: 00000000-001d-9b40-a800-000000479800
session id: 1731c081-07a3-1022-87d2-080069056a64
media id: 1731c083-07a3-1022-87d2-080069056a64

examine this dump?
1: skip
2: restore (default)
 -> 2
this dump selected for restoral

 ------------------------------ end dialog -------------------
--------------

 ======================     subtree     selection     dialog
==========================

the following commands are available:
        pwd
        ls [ <path> ]
```

```
        cd [ <path> ]
        add [ <path> ]
        delete [ <path> ]
        extract
        quit
        help

 -> pwd
cwd is fs root

 -> ls
             155 .profile
            3957 Mensaje
          655508 INFORMIXTMP/
             947 backup.total.ag25
             969 .sh_history
             154 .login
          655499 lib64/
          655490 lib32/
          657696 .desktop-dsi15/
            3963 .expertInsight
             .......... SE SUPRIMEN LINEAS POR EFECTOS DE
LONGITUD
          ..........
            3947 unix
          657728 swap/
          524422 sbin/
             131 hw/

 -> cd swap

 -> ls
          657729 swap1

 -> cd ..

 -> pwd
cwd is fs root

 -> cd etc

 -> ls
          262410 rfind.aliases
```

```
          262310 prioinfo
          409345 config/
          262403 project
          262328 uncompvm
          262431 profile
          262428 gettydefs
          262304 nvram
          ..........  SE  SUPRIMEN  LINEAS  POR  EFECTOS  DE
LONGITUD
          ..........

  -> quit

 ------------------------------ end dialog --------------------
-------------
```

Los subcomandos a nivel del diálogo de la forma interactiva son :

pwd Indica el nombre del directorio donde se encuentra ubicado .
ls Lista el contenido de un directorio almacenado en la cinta .
cd Cambia de directorio en la cinta .
add Adiciona un directorio o archivo a la lista de elementos a extraer .
delete Elimina un directorio o archivo de la lista de elementos a extraer.
extract Empieza la recuperación de los elementos seleccionados .
quit Salir .
help Mostrar la ayuda

Para efectuar la recuperación de la información se podría haber invocado el comando sin la opción **-t** y dentro del diálogo seleccionar la información a recuperar , para luego dar **extract** .

También se podría recuperar en forma normal , con la opción –r , sin entrar al diálogo del modo interactivo .

3.5. COMANDO BACKUP Y RESTORE.

Con estos comandos se pueden sacar backup totales (todo el sistema) o parciales (directorios) , que pueden en un momento dado ser leidos desde el System maintenance Menu (Menu al inicio del arranque del sistema), con la opción Recovery .

La sintaxis de los comandos son :

Backup –h nombre–del–host **–t** nombre–dispositivo nombre–directorio

Restore –h nombre–del–host **–t** nombre–dispositivo

La opción **–h** es en el caso de estar utilizando el dispositivo de backup de otra máquina en la red . Se puede especificar de las siguientes formas :

username@nombre–maquina :nombre–dispositivo

nombre–maquina :nombre–dispositivo

Ejemplo : Hacer un Backup en la unidad /dev/rmt/tps1d7 de la máquina PSA , del directorio /usr/people/hgt :

```
#Backup –h PSA:/dev/rmt/tps1d7 /usr/people/hgt
```

Si se deseara sacar un backup total de la información contenida en todos los discos de la máquina , por la unidad default se daría :

Backup /

Con este Backup , en caso de daño en algún disco , en especial el del sistema operativo , se podría recuperar la información . desde el modo mantenimiento .

Si se desea listar el contenido de estas cintas se utiliza el comando :

List_Tape -h nombre-del-host **-t** nombre-dispositivo

3.6. MANEJO DE BACKUPS INCREMENTALES

Estos pueden ocupar menos espacio y tiempo que los totales . Un esquema de realización de backups incrementales para un sistema de archivos , puede ser :

- El primer día , sacar un baclup total del sistema de archivos completo (podría ser mensual).
- Del segundo al séptimo día , sacar backup de los archivos que han sido modificados el día anterior o backups diarios .
- El octavo día , sacar backup de los archivos modificados en la semana anterior o bacups semanal.
- Repetir los dos pasos anteriores por tres o cuatro semanas .
- Al mes repetir el primer paso .

3.6.1. BACKUP INCREMENTALES CON BRU.

Utilizando las opciones propias del comando :

- Crear un backup completo del sistema de archivos /usr (backup mensual , realizado el 14 de Noviembre)

```
#bru -c /usr
```

- Crear backups diarios , por una semana (desde el 15 al 21 de Noviembre):

```
#bru -c -n 15-Nov-1998 /usr
#bru -c -n 16-Nov-1998 /usr
#bru -c -n 17-Nov-1998 /usr
#bru -c -n 18-Nov-1998 /usr
#bru -c -n 19-Nov-1998 /usr
```

```
#bru -c -n 20-Nov-1998 /usr
#bru -c -n 21-Nov-1998 /usr
```

- Cada semana crear backup semanales (se realizaría el 22 de noviembre y en este caso se involucran los archivos o directorios que han cambiado desde el 15 de noviembre) .

```
#bru -c -n 15-Nov-1998 /usr
```

- Este proceso se repite por tres semanas y al final se repite el backup total mensual .

3.6.2. BACKUP INCREMENTALES CON TAR Y CPIO

Aunque estos comandos en sí , no tienen mecanismos para backups incrementales, se pueden acompañar con otros para la realización de los mismos . Presentaré la forma de hacerlo , con el mismo esquema del caso anterior (la primera linea es la implementación con tar y la segunda linea es con cpio) :

- Cambiarse al sistema de archivos al que se le realizara el backup .

```
#cd /usr
```

- Crear un backup completo de ese sistema de archivos.

```
tar cv

cpio -oLp .
```

- Cada día , sacar los backups diarios .

```
find /usr -mtime 1 -print | tar cvf -

find /usr -mtime 1 -print | cpio -pdL
```

- Cada semana , los backups semanales .

```
find /usr -mtime 7 -type f -print | tar cvf -
```

```
find /usr -mtime 7 -type f -print | cpio -pdL
```

- Cada mes repetir el primer y segundo paso .

3.7. BACKUP AUTOMATICOS

Se puede utilizar la utilidad **cron** , para programar la ejecución de los mismos periodicamente . La cinta debe estar montada en la unidad y es aconsejable que todos los datos se puedan acomodar en la cinta para no requerir el cambio de la misma y así no tener intervención manual .

La siguiente línea debe ser incluida en al archivo : **/var/spool/cron/crontabs/root :**

```
0 3   *  *   *   bru -c -f /dev/rmt/tps1d6  /usr
```

En este caso se programa un backup completo de /usr , todas las mañanas a las 3 :00.

3.8. COMANDO MT

Es usado para dar órdenes y tener más interacción con la unidad de grabación. Permite la manipulación de la cinta magnetica en tareas como rebobinar , avanzar hacia adelante, atrás , definir tamaño de bloque, etc. Es utilizado junto con los comandos tar , cpio , bru para el manejo de múltiples cintas lógicas en una física .

La sintaxis general es :

```
mt  [ -f dispositivo ] comando  [contador]
```

Donde el dispositivo representa el archivo que identifica la unidad de grabación , tal como /dev/rmt/tps1d7 . En las situaciones donde se desee desplazarse sobre las cintas lógicas , es decir avanzar hacia adelante varias posiciones , o regresarse , se debe utilizar el dispositivo que no rebobina

, para evitar que una vez efectuado el posicionamiento , la unidad rebobine la cinta y realmente no la deje en la posición deseada . El comando representa las órdenes que le daremos a la unidad , tal como rebobinar , desplararse hacia adelante , etc . El contador representa la cantidad de posiciones que se debe desplazar la unidad , en los comandos que así lo requieran .

Dentro de los comandos más usados se encuentran :

- **rewind** : Permite rebobinar la cinta y dejarla en el BOT , para iniciar alguna tarea sobre ella .
- **fsf** : Se desplaza hacia adelante en la cinta , tantas posiciones lógicas se le especifiquen en el contador .
- **bsf** : Se desplaza hacia atrás en la cinta , tantas posiciones lógicas se le especifiquen en el contador .
- **status** : Muestra por pantalla información del estado de la cinta . No es necesario que la unidad tenga una cinta cargada .
- **blksize** : Despliega el tamaño de bloque recomendado para I/O, el cual es usado por tar , cpio , bru , etc . El tamaño máximo, mínimo y actual es reportado y estos pueden ser iguales si el drive no soporta tamaños de bloque varialbles .
- **setblksz** : Define el tamaño del bloque a usarse . La mayoría de drives que soportan tamaños de bloque variables , también soportan el uso de diferentes valores , cuando trabajan en modo de bloque fijo . El tamaño permanece definido hasta que la siguiente cinta sea cambiada , o hasta que el drive se use en modo variable .

Ejemplos :

- Ver el estado de una unidad de cinta :

```
mt -t /dev/rmt/tps0d7 status

Controller: SCSI
Device: ARCHIVE: Python 01931-XXX5.56
Status: 0x202
Drive type: DAT
```

Media : Not READY

En este caso la cinta no está colocada en la unidad.

- Para grabación de varias cintas lógicas podriamos realizar el siguiente procedimiento :

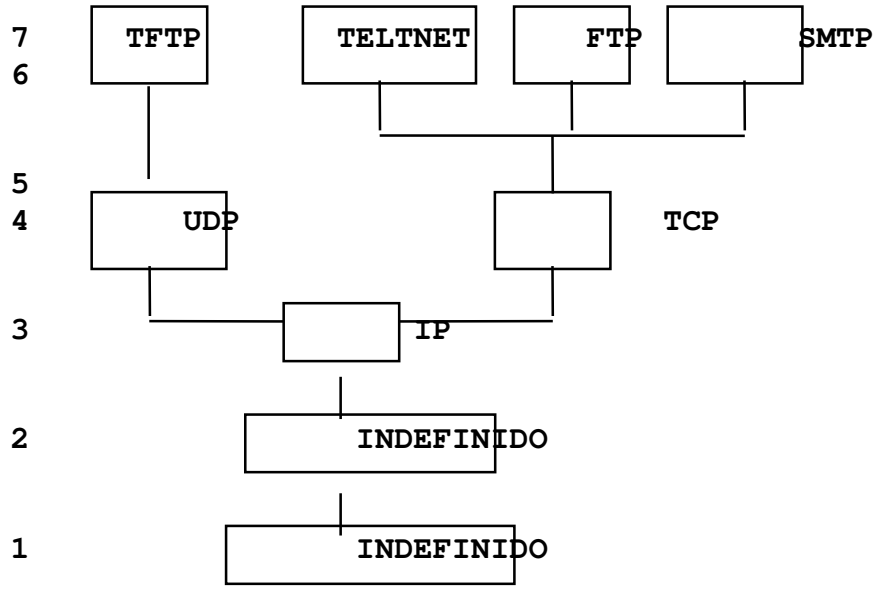
- Para recorrer y visualizar la información almacenada en una cinta lógia :

4. PROCEDIMIENTOS PARA DISCOS

5. TCP/IP

TCP/IP es un producto de comunicaciones que permite conectar Hosts que corran el Tansmission Control Protocol (TCP) y el Internet Protocol (IP) sobre una Local Area Network (LAN) . Fue desarrollado por el Departamento de Defensa de los Estados Unidos para uso militar y adaptado como un estándar para comunicaciones a través de LAN.

Su arquitectura según el modelo ISO/OSI, se define así:



Los dos primeros niveles (1 y 2 que corresponden a Nivel Físico y de Enlace respectivamente), se declaran indefinidos,

para dejar una elección libre del medio de transmisión de datos, sin que esto afecte los demás niveles ni las características del TCP/IP.

Para comprender el modelo anterior necesitamos los siguientes conceptos:

- Internet Protocol (IP): Es un protocolo que provee un servicio de Datagrama, que organiza los datos en paquetes, enruta estos y reensambla los datos al llegar a su destino. En un servicio de datagrama los mensajes son pasados al host en el orden de llegada sin chequeo de errores. Cada datagrama es tratado como una unidad y necesita una dirección en cada paquete.

- Transmission Control Protocol (TCP): Es un protocolo a nivel de transporte, encargado de realizar las conexiones entre host. Registra la secuencia de los paquetes, realiza retransmisiones y otros.

- User Datagram Protocol (UDP): Es usado para enviar y recibir información entre host y soportar transferencia de archivos por TFTP. Soporta la transmisión de segmentos de datos sin secuencia.

- TELNET: Este servicio permite conectar dos sistemas, emulando las terminales propias de cada uno.

- File Transfer Protocol (FTP): Transferencia de archivos entre dos Computadores.

- Simple Mail Transfer Protocol (SMTP): Transferencia de mensajes.

- Trivial File Transfer Protocol (TFTP): Permite intercambiar archivos, pero no es tan completo como el FTP.

En el caso de una red local, debemos tener en cuenta el siguiente diagrama para los tres primeros niveles :

- Address Resolution Protocol (ARP): Es usado para trasladar la dirección Internet en una dirección física de red y viceversa. La dirección física es de 48 bit y no tiene ningún significado para IP . La dirección Internet es de 32 bit y es usualmente escrita en grupos de 8 bits, cada uno de estos en decimal; por ejemplo :

  129.122.64.9 representa
  10000001 01111010 01000000 00001001

La dirección Internet contiene tres elementos :

- Numero de la red.
- Numero de la subred.
- Dirección Local o del sistema.

Hay tres clases de direcciones :

- Clase A : Direcciones de 1-126 ; usan solo el primer octeto para la dirección de la red.

- Clase B : Direcciones entre 128.1 y 191.254 ; usan los dos primeros octetos para la dirección de la red.

- Clase C : Direcciones entre 192.1.1 y 223.254.254 ; usan los tres primeros octetos para la dirección de la red.

5.1. ARCHIVOS DE CONFIGURACIÓN

El papel del administrador es configurar la red, creando o modificando los archivos que mencionaremos a continuación:

/etc/hosts Contiene los nombres y las direcciones Internet de los Hosts involucrados. Para cada host se debe dar:
- Dirección internet.
- Nombre oficial del host.

- Alias o nombre con el que vamos a referenciar el host. El alias para el host local es seguido de la palabra "Localhost".Esta lista no debe incluir todas las máquinas, si se esta usando DNS.

/etc/hosts.equiv Permite a los usuarios de sistemas remotos entrar al sistema sin dar password, si se cumple:

- El sistema remoto tiene una entrada en el hosts.equiv.
- El usuario remoto tiene un user-id en el host local.

El remote login (rlogin) usa este archivo para autenticar usuarios.Esto afecta a todos los usarios de la máquina. Si se desea personlizar se debe recurrir al siguiente archivo.

.rhost Permite a usuarios individuales de sistemas remotos, entrar al sistema sin dar password, cuando el host remoto no tiene una entrada en el archivo hosts.equiv. El usuario necesita colocar un .rhosts en su directorio de trabajo. Este archivo es usado por el "rlogin -l " y otros comandos. El formato del archivo es: (por cada línea)

hostsname user-id

Donde hostname es el nombre del host que se desea accesar y el respectivo nombre de usuario para ese host.

Existen otros archivos que el sistema crea automáticamente, y se pueden observar; que detallaremos a continuación:

/etc/protocols Protocolos usados por aplicaciones corriendo sobre TCP/IP.

/etc/services Servicios disponibles.

/etc/inetd.conf Especifica los procesos y demonios que arrancaran con la red.

/etc/resolv.conf Configuración de DNS. Especifica el dominio y la máquina que actua con servidor de nombres. Además el órden en que sé resolveran los nombres. Ejemplo:

```
hostresorder local bind
domain     psa.edu.co
nameserver 207.24.18.5
```

5.2. COMANDOS INVOLUCRADOS.

Si se desea verificar que la dirección IP y otras caracteristicas fueron definidas apropiadamente para la tarjeta de red, se puede usar los comandos:

```
#netstat -i

Name Mtu   Network         Address            Ipkts Ierrs    Opkts Oerrs  Coll
ef0  1500  200.25.18       pelicano        13033995  1302   964346    39 32492
lo0  8304  loopback        localhost        1065823     0  1065823     0     0


#ifconfig ef0
ef0:
flags=1c63<UP,BROADCAST,NOTRAILERS,RUNNING,FILTMULTI,MULTICAS
T,CKSUM>
     inet    200.25.18.4    netmask    0xffffffc0    broadcast
200.25.18.63
```

Con el netstat se puede observar el nombre de la interface de red. En este caso se llama ef0. Además información sobre paquetes entrantes, salientes y colisiones.

Con el comando ifconfig, se puede observar y modificar los parámetros cargados a las interfaces de red.

El comando netstat tiene otras opciones:

netstat [–a]	Muestra información de todos los servers activos.
netstat [–n]	Muestra las direcciones internet .
netstat [–p] prot	Da información solo del protocolo especificado . Este puede ser : tcp , udp o ip .

Existe un comando que permite revisar la conexión de los hosts, enviando una serie de caracteres al otro equipo y esperando recibir estos de nuevo. Es el comando "ping", cuyo formato es:

ping nombrehost

Si solo se da el nombre del host, se envía una paquete de caracteres y se espera recibir estos nuevamente, en forma indefinida hasta que el usuario detenga la emisión. Se presentan unas estadísticas sobre los caracteres perdidos para estudiar el estado de la conexión y del medio físico. Se puede definir el tamaño del paquete enviado y la cantidad de veces que se realiza esta operación.

5.3. COMUNICACIÓN CON OTROS EQUIPOS UNIX

Los equipos con sistema operacional UNIX,conectados por medio de TCP/IP, soportan además , las llamadas utilidades Berkeley (utilities BSD) :

- **Remote Copy (RCP):** Para copiar tanto archivos como directorios entre dos hosts en red.

  Para copiar un solo archivo:

  rcp [nodofte:archfte] [nododest:archdest]

  Para copiar varios archivos:

rcp [nodofte:]arch1 ...[nodofte:]archn [nododest:]directorio

Para copia de directorios:

rcp –r [nodofte:]direcfte [nododest:]direcdest

- **Remote Login (rlogin)**: Permite hacer login en el host remoto. Se puede tener:

  Login bajo el mismo user–id : entrar al host remoto con el mismo user–id con el que trabajamos en el host local. El comando es:

  rlogin nombre–host–remoto

  Login con diferente usuario ; el comando es :

  rlogin nombre–host–remoto –l user–id–remoto

- **Remote Shell (remsh)**: Permite correr comandos del host remoto. Se pueden tener varias combinaciones, como:

  Ejecutar un comando:

  remsh nombre–host–remoto comando

  Ejecutar proceso local y remoto:

  comando–local | remsh nombre–host–remoto comando–remoto

  Ejecutar comando remoto y local:

  remsh nombre–host–remoto comando–remoto | comando–local

5.4. COMUNICACIONES CON OTROS SISTEMAS OPERACIONALES

Se puede comunicar con otros equipos de diferente sistema operacional, que corran TCP/IP.Los servicios prestados son:

- **TELNET :** Permite hacer login en un host remoto.

  La sintaxis del comando es la siguiente:

  Telnet [nombre-host]

  Si solo Telnet es digitado, el sistema lo situara en modo comandos, cuyo prompt es " Telnet> ". En este modo se pueden dar los siguientes subcomandos:

Open nombre-host	Para conectarse a un sistema.
close	Cierra la conexión.
quit	Salir de TELNET.
help	Ayuda .
status	Muestra información de la conexión actual .

  En el momento que la conexión es hecha, se pasa a modo de transferencia de datos y se interactua con el hosts remoto para realizar Login.

- **FILE TRANSFER PROTOCOL (FTP):** FTP permite realizar transferencia de archivos EXL y otros equipos que corran TCP/IP. La sintaxis del comando es:

  ftp [opciones] [nombre-host]

  donde la opciones pueden ser:

  -d Habilita debug

  -g Deshabilita el globbing (uso de wildcards)

-i Habilita prompt interactivos, es decir,pedir confirmación para cada archivo al trabajar con wildcars.

-n No pregunta por un usuario al establecer la conexión.

Este comando nos coloca en un subsistema cuyo prompt es "ftp>", en donde disponemos de los siguientes subcomandos:

! [comando] Ejecutar comando del sistema local . Sale al prompt de UNIX, donde se regresa con "exit".

append arch [arch-rem]	Agrega el contenido del archivo local (arch), al archivo remoto (arch-rem).
type ascii	Pasa el tipo de archivo a ASCII.
type binary	Pasa el tipo de archivo a IMAGE.
type image	Pasa el tipo de archivo a IMAGE.
bell	Activa o desactiva sonido de campana al terminar copia de archivos.
bye	Termina FTP y la sesión remota.
cd directorio	Cambia el directorio de trabajo del host remoto .
close	Termina sesión remota y permanece en FTP.
debug [on/off]	Al activar debug (on), cada comando enviado al sistema remoto es mostrado precedido por el string -->
delete archivo	Borra archivo en el host remoto.

dir [direct] [archivo] Lista el contenido del directorio del host remoto y sitúa este en el archivo del host local .

get arch-rem [arch-local] Transfiere archivo del host remoto al local.

glob Interpretación de wildcards. Si esta en "on", se pueden usar estos; si es "off", los wildcars son interpretados literalmente .

hash Impresión del símbolo "#", por cada bloque de datos transmitido (1024 bytes).

lcd directorio Cambia el directorio de trabajo del host local.

ls [directorio [archivo]] Muestra información abreviada del contenido del directorio remoto y sitúa esta en un archivo del host local.

mdelete archivos Borra archivos del host remoto.

mkdir directorio Crea un directorio en el host remoto.

mget [archivos] Transfiere archivos, usando wilcards desde el hosts remoto.

mput [archivos] Envía archivos al host remoto, usando wilcards.

open nombre-host Establece conexión con el host remoto.

prompt Si esta "off", múltiples archivos son procesados sin pedir confirmación para cada uno.

put arch-local [arch-rem] Envía un archivo al host remoto.

pwd	Muestra el nombre del directorio actual del host remoto.
quit	Termina sesión de FTP.
rename nombre1 nombre2	Cambia de nombre al archivo nombre1 del host remoto.
rmdir directorio	Borra directorio del host remoto.
status	Muestra los parámetros para transferencia de archivos .
user [user-id [password]]	Realiza login en el host remoto.

- **TRIVIAL FILE TRANSFER PROTOCOL (tftp):** Ofrece un subconjunto de funciones del ftp. Permite copiar archivos entre sistemas que soporten tftp; no ejecuta autenticacion de usuarios, por esto los archivos deben poderse leer y escribir por todos los usuarios . El formato del comando es:

  Para enviar archivos:

  tftp remotehost put sourcefile [targetfile]

  Para recibir archivos:

  tftp remotehost get sourcefile [targetfile]

  Para entrar a nivel de subcomandos:

  tftp [nombrehost]

  Entramos al servicio, cuyo prompt es "tftp>", donde disponemos de los siguientes subcomandos:

connect [nombrehost]	Realizamos conexión con el sistema remoto.

```
get [nombrehost:]arch-remoto .... arch-local

put arch-local .... [nombrehost:]direc-remto

quit                        /* Salir de tftp .

?                        Ayuda.
```

5.5. NFS (NETWORK FILE SYSTEM)

NFS es un servicio de red que permite a los usuarios accesar los sistemas de archivos y directorios de otros equipos en la red. Los Hosts pueden ser de diferente marca y sistema operacional. Estas diferencias son transparentes para los usuarios.

Se debe tener en cuenta los siguientes conceptos:

Servidor: El hosts que permite el acceso a sus sistemas de archivos o directorios locales (este debe exportar sus recursos).

Cliente : El hosts que solicita el acceso a sistemas de archivos o directorios de otras máquinas (este debe montar los recursos exportados por otras máquinas).

5.5.1. COMANDO EXPORTFS.

Permite poner a disposición de otros host, sus sistemas de archivos o directorios locales.

Este comando arranca leyendo el archivo **/etc/exports**. Si el archivo exports es modificado, se debe volver a correr comando exportfs para actualizar. El server mantiene un registro de recursos exportados que son actualmente montados y los nombres de los clientes que los están utilizando (archivo **/etc/rmtab**). La información de este archivo es vista con el comando **showmount.**

Opciones Básicas.

-a Exporta todos los recursos listados en /etc/exports.

-u Termina el export del recurso escrito a continuación.

-v Muestra mensajes de salida durante el proceso.

Archivo /etc/exports. El formato de este archivo es:

pathname opciones

Donde :

pathname: Es el sistema de archivos o directorio que se va a colocar a disposición de los otros host.

opciones: ro Solo lectura
rw Lectura y escritura
access Da accesos para una determinada lista de clientes solamente.

Ejm:

```
/   -ro
/usr/demos  -ro,access=cliente1:cliente2:cliente3
/usr/catman
```

COMANDO MOUNT Y UMOUNT.

Después que el servidor a colocado a disposición sus recursos, desde la máquina cliente se deben montar estos. Después de utilizados, se deben desmontar; para lo cual existen los comandos mount y umount respectivamente. Las opciones básicas de estos son:

-t Define el tipo a ser montado (Debe ser **nfs**).

-a Intenta montar todas las entradas listadas en el archivo /etc/fstab, o desmontar las encontradas en el archivo /etc/mtab.
-h Intenta montar o desmontar las entradas, para un host en particular.
-o Lee las opciones desde la línea de comando y no desde el archivo.

Archivo /etc/fstab. El formato de este es:

file-sytem mount-point type options frec pass

Donde:

file-system: Directorio remoto del servidor a ser montado.

mount-point: Directorio local donde se montará el recurso.

type: Debe ser nfs.

options: Pueden ser:

ro	Solo lectura
rw	Solo escritura
hard	En caso de falla, el cliente espera a que el servidor responda.
soft	En caso de falla, el cliente hace determinado número de intentos.
bg	Permite el mount en backgroun.
fg	Permite el mount en foreground
noauto	Al hacer mount, ignora esta entrada.

frec: Debe ser 0 , para este caso.

pass: Debe ser 0 , para este caso.

Ejm:

hgt:/usr/people /usr/usuarios nfs ro,hard,bg 0 0

ACTIVANDO EL SERVIDOR.

Se asume que el software ya esta instalado. Se debe tener en cuenta los siguientes pasos:

- Chequear que el nfs esta activado con el sistema:

```
# chkconfig
```

Debe estar "on". Si no lo esta, activarlo y rebootear la máquina:

```
# chkconfig nfs on
```

- Verificar que los daemons esten corriendo (nfsd, biod).

-Verificar que el daemon del mount este registrado por el portmaper:

```
# /usr/etc/rpcinfo -p | grep mountd
```

-Verificar el archivo /etc/exports

-Exportar los recursos:

```
# exportfs -av
```

-Verificar:

```
# exportfs
```

ACTIVANDO EL CLIENTE (SGI).

Se asume que el software ya esta instalado. Se deben tener en cuenta los siguientes pasos:

-Chequear que el nfs esta activado con el sistema:

```
# chkconfig
```

Debe estar "on" . Si no lo esta, activarlo y rebootear la máquina:

```
# chkconfig nfs on
```

-Editar y organizar el archivo /etc/fstab

-Crear el mount-point:

```
#mkdir /usr/usuarios
```

-Montar el recurso, bien sea manualmente, o procesando el /etc/fstab.

Ejm:

```
# mount -a

# mount hgt:/usr/people  /usr/usuarios
```