Taller de Unix

Gilberto Díaz

29 de septiembre de 2010

Licencia de Uso

Centro de Teleinformación
Corporación Parque Tecnológico de Mérida
Mérida Edo. Mérida - Venezuela

Índice General

Índice de Figuras

Índice de Tablas

Acerca de este manual

Audiencia

Este manual, al igual que el curso, está dirigido a personas que han tenido muy poca, o ninguna, experiecia con sistemas operativos compatibles con Unix/Linux. Asi mismo, los introduce en los detalles de la administración de este tipo de sistemas operativos. Intenta igualmente ser un tutorial organizado por tipo de tarea.

Objetivos

Este material trata sobre el uso, manejo y administración del sistema operativo Unix Incluye ejemplos prácticos basados en GNU/Linux

Al finalizar este manual usted debe estar en capacidad de:

- Acceder a su ambiente de trabajo en una máquina GNU Linux.
- Entender y utilizar los distintos elementos del sistema operativo.
- Entender y utilizar los conceptos de directorios y archivos.
- Ejecutar aplicaciones.
- Utilizar aplicaciones de red para comunicarse con otros sistemas
- Instalar sistemas Unix basados en Linux.
- Conocer los detalles de arranque y detención de Unix.
- Manejar usuarios.
- Administrar sistemas de archivos.
- Administrar procesos y aplicaciones.
- Administrar interfaces de red.
- Manejar sistemas de impresión.

- Conocer los diferentes detalles de seguridad.
- Conocer herramientas para supervisión de redes.

Organización

Este manual está organizado en 5 capitulos como sigue:

- **Introducción al Sistema Operativo Linux** Aquí se hace un recuento de la historia de este sistema operativo y se introducen algunos conceptos básicos y útiles para entender la relación de los sistemas operativos y las computadoras.
- **El ambiente del Usuario** Es una introducción a los ambientes de trabajo de los usuarios. Aquí se describen los interpretadores de comandos o *conchas* y los ambientes de ventanas de GNU Linux, en particular el ambiente **X**
- **Manejo de Archivos** Muestra el funcionamiento y comandos básicos del editor de archivos de Linux `vi`. Se hace, también, una introducción a algunos otros editores de archivos.
- **Manejo de Directorios** En esta seccion se demuestra la forma en la que el usuario debe ver el contenido y propiedades de los directorios ademas de crear nuevos directorios asi como tambien eliminarlos, en el mismo orden de ideas el usuario podra a su vez ser capaz de buscar archivos y manejar la permisologia de los mismos.
- **Manejo de medios de almacenamiento secundarios** Se muestra el uso correcto de los distintos medios de alamacenamiento secundarios como Pen Drives y Discos Compactos, ademas de compartir los Directorios en red .

Información relacionada en el Web

En el web existen los siguientes documentos:

- A Basic UNIX Tutorial:
  http://www.isu.edu/departments/comcom/unix/workshop/unixindex.html
- Linux Online Courses
  http://www.linux.org/lessons/
- Introduction To UNIX:
  http://www.ceas.rochester.edu:8080/CNG/docs/IntroUnix.html
- Unix
  http://metalab.unc.edu/echernof/unix/what.html

Convenciones

En este manual se utilizan las siguientes convenciones:

`%`	Un signo de porcentaje al iniciar una linea en los ejemplos representa el *prompt* (mensaje de espera) de una concha tipo **C shell**
`$`	Un signo de dollar representa el *prompt* de una concha tipo **Bourne Shell**.
`#`	Un signo de número representa el *prompt* de superusuario.
`ctrl-x`	La secuencia de caracteres `ctrl` delante de una letra indica que se debe mantener presionada la tecla **Ctrl** al mismo tiempo que se presiona la letra indicada.
`alt-x`	La secuencia de caracteres `alt` delante de una letra indica que se debe mantener presionada la tecla **alt** al mismo tiempo que se presiona la letra indicada.

Capítulo 1

Introducción

"El número de instalaciones de UNIX se ha elevado a 10, y se espera que aumente."
Brian Kernighan - El entorno de programación UNIX

¿Qué es Unix? Unix es un sistema operativo que controla los recursos de una computadora y la interacción del usuario con esta. Permite a uno o varios usuarios correr sus programas y controla los dispositivos externos, como cintas, impresoras, terminales, etc. Por la forma como maneja los recursos de la máquina y la política de tiempo compartido entre procesos, el Unix es un sistema operativo multiusuario.

El sistema operativo Unix fue desarrollado en una máquina desechada (DEC PDP-7) en los laboratorios Bell durante 1969 por Ken Thompson, Rudd Canaday, Doug McIlroy, Joe Ossanna y Dennis Ritchie. Fue desarrollado como un sistema de uso general lo bastante adecuado y comodo como para atraer usuarios entusiastas. Esto justificó la adquisición de una nueva máquina, la PDP-11-20, para continuar su desarrollo en 1970.

En el año 1973, Ritchie y Thomson reescribieron el *kernel* (núcleo) del Unix en C, en contra de la tradición de escribir el sofware de sistema en lenguaje ensamblador. Con esta reescritura el Unix adquirió básicamente la forma que le conocemos hoy.

El año 1974 se introdujo Unix en alguna Universidades con "fines académicos" y al cabo de pocos años ya estaba disponible comercialmente. En este tiempo ya el Unix esta en manos de los desarrolladores de software, tales como desarrolladores de procesadores de palabras, sistemas de apoyo de operaciones en compañías, etc.

El gran triunfo de Unix se debe a que está escrito en C y basado en un núcleo muy pequeño, lo que le da oportunidad de ser instalado y correr de forma eficiente en toda la gama de computadores existentes en el mercado desde los *Palm Top* hasta los supercomputadores, además su código fuente está disponible y escrito en un lenguaje de alto nivel, lo que lo hace más fácil de adaptar a las exigencias más particulares. Por último es un sistema operativo que establece un exelente ambiente para programadores, ya que es muy estable y tiene una enorme cantidad de herramientas que aumentan su productividad.

" Hello everybody out there using minix - I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones..."
Linus Torvalds

Al inicio de la década de los noventa el sistema operativo predominante en el mundo de los computadores personales era MSDOS (Microsof Disk Operating System). Las computadoras Apple constituian una alternativa poco accesible debido a sus precios elevados.

Otro de los sistemas operativos importantes en esa época era Unix, desarrollado en los laboratorios Bell durante 1969 por Ken Thompson, Rudd Canaday, Doug McIlroy, Joe Ossanna y Dennis Ritchie. El año 1974 se introdujo Unix en alguna Universidades con "fines académicos" y al cabo de pocos años ya estaba disponible comercialmente. Sin embargo, los proveedores mantuvieron los precios elevados de tal manera que Unix se mantuvo lejos del ambiente de los PCs. La solución parecía ser MINIX, desarrollado por el profesor Andrew Tanenbaum para la enseñanza a sus estudiantes. Este sistema operativo no fue exitoso como tal pero existía la ventaja de que su código fuente estaba disponible al público.

GNU Linux es un *clon* del sistema operativo Unix, inspirado en MINIX, de libre distribución, diseñado originalmente para máquinas 80386 y 80486. El sistema operativo GNU Linux fue desarrollado por Linus Torvalds en la Universidad de Helsinki - Finlandia en 1991. La versión 0.01 fue liberada por Linus en Septiembre de ese año.

El movimiento GNU ha generado una gran cantidad de aplicaciones y programas que realizan diferentes tareas y están dedicadas a diferentes propósitos. Organizaciones como Red Hat, Mandrake, SUSE, Debian, Gentoo, etc, recopilan su propio conjunto de esas aplicaciones, modifican el núcleo y elaboran sus propios programas de instalación. Esta compilación de aplicaciones, núcleo e instaladores conforman lo que se conoce como **distribución**.

1.1. Características del Sistema Operativo

El sistema opetativo Unix presenta una serie de características que lo hace atractivo a los usuarios y adecuado para desempeñarse en la mayoria de las áreas importantes de la computación. Está ampliamente difundido en máquinas que fungen como servidoras de base de datos, procesamiento de información, almacenamiento, etc.

1.1.1. Tiempo Compartido

Esta característica es la base de mucha de las propiedades de Unix. El tiempo compartido consiste en una cola de procesos listos para entrar al procesador y ser ejecutados. A cada proceso se le asigna un lapso de tiempo determinado, denominado **quantum**, para permanecer dentro del procesador. El proceso abandona el procesador cuando finaliza o

cuando su quantum ha expirado. En este último caso el proceso regresa a la cola para esperar de nuevo por el procesador.

1.1.2. Multitarea

Gracias a la existencia de una cola de procesos listos, se puede cargar más de una tarea en memoria, y entrar a esta cola para competir por el procesador. La ejecución de procesos se hace de forma secuencial, sin embargo, la rapidez con que se mueve el flujo de procesos en el sistema crea la ilusión de ejecución concurrente de trabajos.

1.1.3. Multiusuario

Unix es capaz de distinguir entre diferentes usuarios que entran y utilizan el sistema. Esto significa que dos o más personas pueden ejecutar tareas simultáneamente en el mismo procesador. Para establecer la diferencia entre los distintos usuarios se le asigna un nombre, **login**, a cada uno de ellos.

1.1.4. Núcleo Codificado en Lenguaje de Alto Nivel

Uno de los creadores de este sistema operativo fue Dennis Ritchie, el creador del lenguaje C. En sus inicios, los desarrolladores transcribieron el núcleo al lenguaje C. Esto ha sido uno de los grandes logros de Unix ya que por esta razón es muy fácil trasladar el sistema operativo a cualquier plataforma.

1.1.5. Diseñado Originalmente para Programadores

Originalmente Unix estaba dirigido a un tipo de usuarios particular, programadores. Por lo tanto, cuenta con una serie de herramientas adecuadas para el desarrollo de programas. Actualmente, Unix está capacitado para dar soporte a un número considerable de áreas, como por ejemplo, bases de datos, servicios de red, control de sistemas, comunicaciones, visualización, cálculo intensivo, etc.

1.2. Componentes

1.2.1. Hardware

Unix ha sido trasladado a la gran mayoría de las plataformas actuales, desde las computadoras de mano (Agendas electrónicas) hasta las máquinas más grandes del planeta (supercomputadoras). Esto ha sido posible gracias a que su núcleo ha sido codificado en lenguaje de alto nivel.

1.2.2. Núcleo

El núcleo conforma el corazón del sistema operativo. Este es un procesos que siempre se encuentra en memoria principal y es el encargado de controlar el hardware, administrar los procesos, asignar memoria a los procesos, gestionar los dispositivos periféricos de la máquina, etc.

1.2.3. Conchas (Shells)

Los interpretadores de comandos, o conchas, conforman la interfaz mediante la cual los usuarios pueden comunicarse con el sistema operativo y ejecutar sus requerimientos. Estos son programas comunes que se encargan de leer, interpretar y ejecutar las intrucciones dadas por los usuarios. En la actuliadad, existen diferentes tipos de conchas, cada una con sus propias características y ventajas.

1.2.4. Programas de Aplicación

Los sistemas Unix actuales constan de una serie de aplicaciones especializadas para ejecutar diversas tareas que van desde el procesamiento de imágenes, pasando por el procesamiento de textos, bases de datos, etc, hasta juegos. En un sistema Unix totalmente instalado, el número de comandos presentes puede alcanzar una cifra de dos mil.

La figura 1.1 muestra la jerarquía de los componentes de un Sistema Unix.

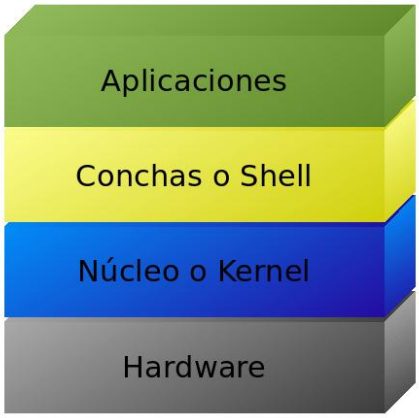

Figura 1.1: Componentes del Sistema Operativo

Capítulo 2

El Ambiente del Usuario

Unix es un sistema multiusuario, por esto, es necesario establecer una configuración personal y un sistema de permisología para cada usuario. Esto permite que el usuario mantenga información confidencial en el sistema y que otro usuario no pueda accederla. Además, cada usuario puede tener una configuración para realizar unas u otras tareas sin afectar el trabajo de otros usuarios. Esta configuración, más la información de la estructura de directorios que pertenecen al usuario, su *login* y su *password*, conforman lo que se llama una **cuenta de usuario**.

Las cuentas de usuario tienen varias utilidades, aún cuando la principal es la de permitir a un usuario entrar en el sistema y correr programas, existen cuentas especiales con privilegios y funciones particulares. La más importante de las cuentas especiales es la del administrador del sistema, esta es llamada *root* y tiene los máximos privilegios, es decir es la única cuenta que puede cambiar la configuración del sistema, instalar programas para todos los usuarios, apagar o reiniciar el sistema y tiene ademas el privilegio de leer todos los directorios del sistema, incluyendo los de los usuarios. Existen otras cuentas especiales como por ejemplo: *nobody* la cual no posee nigún privilegio y es utilizada para iniciar servicios de red que necesitan de cierto nivel de seguridad como lo es el servidor de web, etc. Nobody solo tiene acceso a su propio directorio y algunos comandos muy básicos. Otra cuenta especial es *lp*, en algunos sistemas se designa este usuario virtual para que controle los procesos de impresión.

Actualmente en GNU Linux, al igual que en otros sistemas Unix, se cuenta con distintos ambientes que pueden ser utilizados por los usuarios para realizar sus labores. El ambiente original de trabajo es el interpretador de comandos o *shell* el cual cuenta con una serie de instrucciones que permiten dar indicaciones al sistema operativo para ejecutar diversas tareas. Asi mismo, existen diferentes manejadores de ventanas entre los sistemas GNU Linux. En la actualidad existen una gran variedad de ambientes de ventanas del dominio público como: kde, fvwm, afterstep, enlightenment, etc. los cuales pueden obtenerse de cualquier sitio en internet y ser compilado e instalado en cualquier plataforma GNU Linux.

Todos los ambientes de ventanas conocidos se apoyan en el sistema X Windows. El

sistema X Windows es un estándar introducido por el MIT al mercado del UNIX, que encontró rápido apoyo en compañías como: IBM, SUN, DEC, HP y AT&T. En estos momentos la versión más utilizada y que se instala por omisión en casi todas las plataformas de GNU Linux es la versión 11 o X11. De esta versión se han construido varias reediciones conocidas como X11R5 y X11R6 (las más utilizadas).

El sistema X11 está basado en el modelo cliente-servidor para el despliegue gráfico. En el sistema corre un programa que controla la tarjeta gráfica y que es el servidor gráfico. Los programas que requieren una salida gráfica son los clientes y deben comunicarse con el servidor. Este modelo permite que el programa cliente se ejecute en una máquina remota en la red y el despliegue gráfico se haga en la máquina local.

2.0.5. Entrada al sistema

Una de las características de GNU Linux, al igual que todos los sistemas operativos modernos, es que GNU Linux es *full duplex*, es decir, que al escribir una letra esta aparecera reflejada de inmediato en el monitor. Sin embargo en algunos casos especiales este reflejo es cortado para seguridad del usuario o para asignar a la tecla una función diferente a la normal. Uno de estos casos es cuando se introduce el *password*, que el usuario escribe una serie de caracteres y en el monitor no aparece nada.

La unica forma de tener acceso a un sistema GNU Linux es a través de una cuenta de usuario. Se debe conocer el *login* o nombre de la cuenta y el *password* o clave de acceso.

Luego de introducir el *login* y el *password* ocurrirá un corto período de tiempo de espera, durante el cual el sistema valida la información suministrada. Si se utiliza el terminal sin ambiente de ventanas veremos algo como esto:

```
This is odie.ing.ula.ve (Linux i686 2.6.11.11) 16:28:41

odie login: gilberto
Password:
Last login: Fri Jul 29 16:33:11 on vc/1
gilberto en odie>
```

Esta información muestra generalmente la versión del Sistema Operativo GNU Linux que tiene el sistema y alguna otra información adicional que puede variar de acuerdo al sistema que se este utilizando.

Al ingresar el *login* y el *password* es importante recordar que GNU Linux hace diferencia entre letras mayúsculas y minúsculas, por ejemplo, si el password fue establecido como `Un_3T28aa`, debe ser escrito conservando el orden de letras mayúsculas y minúsculas.

Si utilizamos un ambiente de ventanas, también se le pide al usuario el login y la contraseña. Como ejemplo, utilizaremos el ambiente de ventanas **KDE** y la la solicitud de login y clave se verá como se muestra en la figura 2.1.

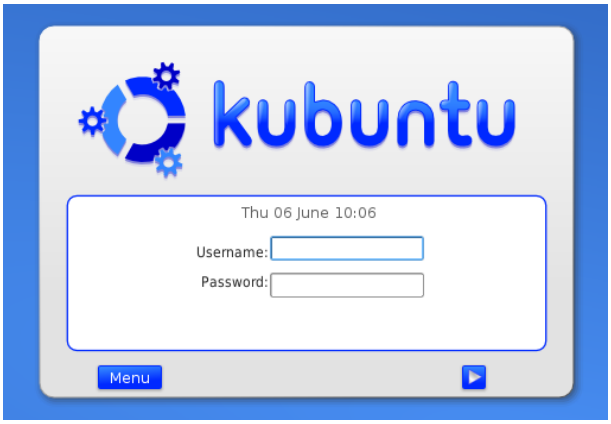

Figura 2.1: Solicitud de Login y Contraseña

Luego de haber ingresado el nombre y contraseña, el sistema cargará el ambiente de ventanas dependiendo del usuario y se mostrará el escritorio principal. Véase la figura 2.2

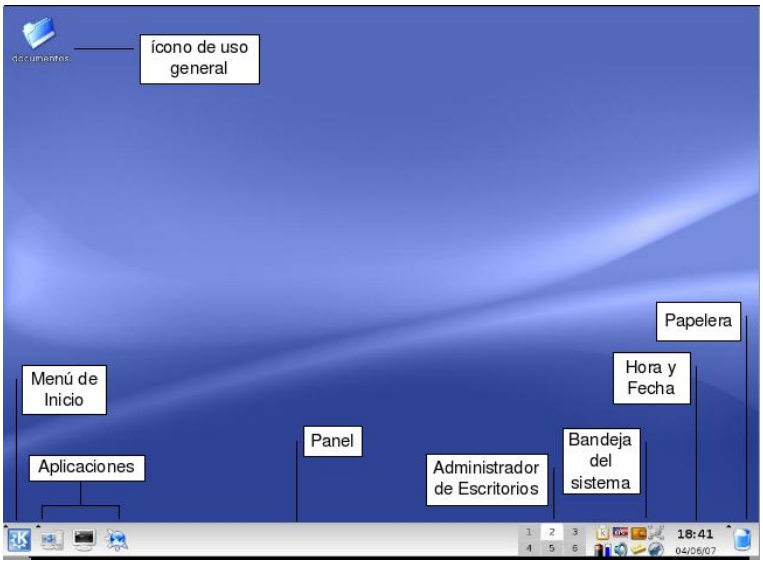

Figura 2.2: Escritorio del Usuario

A continuación se hace una breve descripción de los elementos que podemos encontrar en el escritorio

- **Iconos de uso general.**

  En este se encuentran unos íconos que comunmente son aplicaciones. Estos pueden ser modificados, removidos o se puede agregar otros.

Menú de Inicio: en este se encuentran las aplicaciones, ls configuración del sistema y las sesiones de escritorios.

Menú del Sistema: Al hacer click en este ícono se mostrará el menú de los sitios importantes del sistema.

- **Escritorio.**

  En este podemos encontrar íconos de los accesos directos, archivos, directorios o aplicaciones dependiendo de la configuración que le haya dado el usuario.

- **Panel:**

  Es un componente del ambiente de ventanas, el cual es capaz de de ofrecer aplicaciones útiles de una manera rápida como los accesos directos.

  Gracias a este podemos trabajar con diferentes sitios y aplicaciones simultáneamente, es decir, todas las aplicaciones que esten activas son mostradas como botones en el panel. Con hacer click en el botón de la aplicación esta se maximizará en la pantalla. *Nota:* si se hace click en un aplicación ya maximizada esta automáticamente se minimizará.

- **Administrador de Escritorios.**

  Los ambientes de ventanas para Linux tienen más de un escritorio, con el administrador se puede cambiar de escritorio. Con esta característica el usuario puede organizar mejor todas las ventanas de las aplicaciones que tiene abiertas.

- **Bandeja del sistema**

  En la bandeja del sistema es un pequeño panel donde se encuentran varias aplicaciones que se pueden ejecutar directamente desde el escritorio tales como notas, reproductor de música, acceso a internet, etc. Cada una de las estas aplicaciones comunmente se encuentran en el menú de inicio.

- **Papelera**

  Esta es una localidad en el cual estan ubicados todo aquellos archivos que se haya eliminado y desde alli los mismos pueden ser recuperados.

El icono del menú de inicio es uno de los mas importantes del panel ya que por medio de este se accesa a todas las aplicaciones, sesiones de escritorio y programas del sistema. Al hacer click en el icono se despliega la lista de aplicaciones a continuación (figura 2.3)

Todas las aplicaciones gráficas del sistema se pueden ejecutar utilizando este menú y se encuentran ubicadas en submenus que las agrupan de acuerdo a categorias particulares

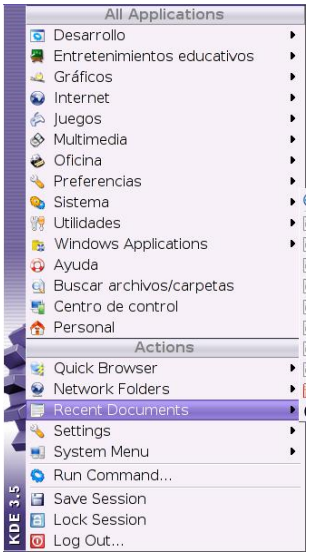

Figura 2.3: Lista de aplicaciones del Menú de Inicio

Capítulo 3

Tareas Básicas

Como hemos mencionado, en GNU Linux, al igual que en otros sistemas Unix, se cuenta con distintos ambientes de trabajo. En este capítulo estudiaremos como realizar tareas comunes tales como: gestión de archivos, gestión de directorios, etc. utilizando interpretadores de comandos y ambientes de ventanas.

3.1. Utilizando Interpretadores de comandos (Shells)

Las conchas o *shells* son los programas de GNU Linux que interpretan los comandos suministrados por el usuario; estas se presentan como una interfaz interactiva basada en texto. La primera concha UNIX, llamada *sh*, era una concha que ofrecía pocas posibilidades de interacción. Con el tiempo se fueron desarrollando conchas más amigables como la *csh*. Actualmente se pueden agrupar de la siguiente manera: dos conchas que utilizan una sintaxis similar a las *csh* (*csh* y *tcsh*) y cuatro que utilizan una sintaxis igual a la de *sh* (*sh*, *ksh*, *bash* y *zsh*). La última generación de conchas (*tcsh*, *ksh*, *bash* y *zsh*) introducen nuevas características como la de editar los comando en línea, la posibilidad de utilizar barras de desplazamiento (*scroll bar*), mecanismos para recuperar comandos anteriores (historia) y comandos para completar nombres (*command/file-name*).

Los *shells* modernos se han convertido en más que un simple interpretador de comandos. Estos *shells* poseen lenguajes de programación con posibilidades de utilizar estructuras elaboradas de decisión y de repetición. Esto permite la elaboración de rutinas o *scripts* basadas en comandos de GNU Linux y las estructuras del *shell* en uso y correrlos como nuevos comandos. Al correr una rutina escrita en el "lenguaje" *shell* se genera una nueva instancia de la concha, esta *subshell* corre el programa y al salir deja el *shell* padre intacto.

A través de los *scripts* se pueden realizar tareas tediosas y habituales con un solo comando.

3.1.1. Tipos de Interpretadores

Básicamente existen dos vertientes de interpretadores de comandos, los *C shell* y los *Bourne Shell*. La diferencia entre ambos es el estilo que se utiliza para las funciones avanzadas como los son la definición de variables de ambiente, los *scripts* y la sintaxis en el lenguaje.

Como se mencionó anteriormente, el *prompt* por omision, que aparece en pantalla depende del tipo de concha que se utilice. Si tenemos una concha del tipo csh, el prompt será %, para las conchas sh o ksh tendremos $, el prompt # está reservado para el administrador del sistema o root. Si aparece un prompt más personal, como por ejemplo el nombre de la máquina, es porque en alguno de los archivos de configuración del usuario hay un comando que permite ponerle algún nombre al aviso de espera.

3.1.2. Archivos de configuración de las Conchas

Los archivos de configuración son básicamente archivos de comandos de GNU Linux que son leídos al iniciar una sesión en uno de los *shell*. En estos archivos se define el ambiente del usuario, que consta de la información sobre los caminos en los que busca los comandos, las variables de ambiente etc.

Para las conchas tipo *C Shell* existen dos archivos de configuración el *.login* y el *.cshrc*, mientras que en las conchas tipo *Bourne Shell* las configuraciones se hacen en los archivos *.profile* y *.kshrc* (si está utilizando **ksh**). El archivo de configuración para bash es .bashrc

Ejemplo de *.bashrc*:

```
# .bashrc

# User specific aliases and functions

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

MACHINE=‘hostname -s‘
USER=‘whoami‘

#PS1="$USER en $MACHINE>"
PS1="\[\033[01;34m\]$USER en \h\[\033[00m\]>"
PATH=/usr/local/mpich/bin:$PATH:/sbin:/usr/sbin:\
~/comandos:/usr/local/netscape:/
alias ls=’ls --color’
alias h=’history’
alias cs=’clear’

export HISTFILESIZE=5000
export HISTSIZE=5000
```

```
export TERM=xterm
```

El significado de cada línea será aclarado al transcurrir el manual.

3.1.3. Características de las Conchas

Las conchas poseen funcionalidades adicionales a la interpretación de comandos. Dentro de esas capacidades se tiene:

- **Ejecución de programas y secuenciamiento de comandos:** Cuando un usuario escribe los comandos, lo hace desde la línea de comandos. En general, ésta está conformada por un comando y sus argumentos. Esta línea es analizada por la concha, la cual es responsable de identificar el comando y de revisar si hay metacaracteres para realizar un procesamiento adicional sobre esos metacaracteres. Luego la concha arranca la ejecución del programa. Cuando esta ejecución termina el control vuelve a la concha. Por otra parte se puede ordenar la ejecución de varios comandos en secuencia en una misma línea de comandos, utilizando el caracter punto y coma ";".

  Ejemplo:

  % **ls;date**

  ```
  PROD3.txt           indice.aux  indice.dvi  indice.tex  prueba.ps
  PROD6.txt           indice.bbl  indice.log  indice.toc  tallerunix
  Sat Nov  6 03:01:23 VET 1999
  %
  ```

  Observe que después del listado de los archivos la fecha actual es mostrada, la cual es la salida del segundo comando.

- **Sustitución de nombres de archivos:** La especificación del nombre de un archivo puede ser generalizada a través del uso de caracteres tales como "*", "?". La concha se encarga de realizar la sustitución de tales caracteres en los nombres.

  Ejemplo:

  % **ls -l archivo?**

  ```
  archivo1 archivo2 archivo3 archivoa archivoz
  %
  ```

  En este caso se indica que para el comando ls serán considerados todos aquellos archivos que empiecen con la palabra archivo y terminen con cualquier otro caracter, tales como archivo1, archivo2, ... archivoa, archivoz, etc.

- **Redirección de entrada/salida:** GNU Linux realiza el tratamiento de todos los componentes de una máquina mediante archivos especiales. Toda concha tiene asignado un dispositivo específico o archivo especial para la salida estándar, la entrada estándar y el despliegue de errores. Las conchas tienen la capacidad de utilizar archivos alternos para manejar la entrada, la salida y el despliegue de errores. Esto se logra a través del redireccionamiento, y para ello se utilizan los caracteres: $>, >>, <, <<, 2>, 2>>$.

  Por ejemplo, existe un archivo especial que maneja la pantalla el cual es designado como la salida estándar y también para el despliegue de errores. Cualquier comando ejecutado enviará su salida a tal archivo. Si se desea guardar esta salida en un archivo convencional, entonces deberá ser ejecutado el comando como sigue:

  % **comando** > **arch**

  Si el archivo arch no existe, será creado y contendrá la salida del comando. Si arch ya existia, entonces será sobreescrito y tendrá como contenido la salida del comando. Si el archivo ya existe y se desea conservar el contenido, se debería ejecutar el siguiente comando:

  % **comando** >> **arch**

  Esto adiciona la salida del comando al final del archivo.

- **Encauzamiento o pipes:** Otra capacidad de las conchas es poder redirigir la salida de un comando hacia otro comando. Este último, tomará como entrada la salida del primero. Para lograr esto, se utiliza el caracter "|" de la forma siguiente:

  % $\mathbf{cmd}_1$ | $\mathbf{cmd}_2$ |**...**| $\mathbf{cmd}_n$

- **Control del ambiente:** La concha permite adaptar el ambiente a las necesidades del usuario, a través de las variables de ambiente tales como: PATH y HOME y además permite modificar el caracter de espera del sistema, prompt (El ambiente puede estar caracterizado por otras variables, manejadas también por la concha).

- **Interpretador de lenguaje de programación:** Las conchas "entienden" un lenguaje de programación. Un código en ese lenguaje puede ser escrito en la línea de comandos o guardado en un archivo para ser ejecutado posteriormente. En las próximas secciones se explicará el lenguaje de comandos de la concha Bash.

GNU Linux tiene dos tipos de comandos, los comandos que forman parte del *shell* y los comandos del sistema. Es por esto que de aqui en adelante haremos uso de las conchas tipo *bash (Bourne again shell)*, ya que los comandos de las *C Shell* son diferentes los cuales se pueden encontrar con el comando **man** (que estudiaremos más adelante), en las páginas `ksh(1)` y `sh(1)`.

Los comandos que forman parte de las conchas *Bash* se muestran en la tabla 3.1.

comando	descripción
`alias`	asigna y muestra una definición de alias
`bg`	Coloca un trabajo suspendido en ejecución de fondo
`echo`	Escribe el argumanto a la salida estandar.
`fg`	Pasa un trabajo, que esté en ejecución de fondo, a ejecución interactiva
`history`	Muestra el contenido de la historia de comandos
`jobs`	Muestra el numero de trabajo y el PID de los trabajos que están corriendo en el fondo
`logout`	termina la sesión de trabajo
`rehash`	Le indica al *shell* que debe recalcular la tabla de *hash*, de modo que pueda encontrar un comando recién instalado
`repeat`	Repite un comando un número específico de veces
`set`	Establece y muestra una variable de la concha
`env`	Establece y muestra una variable de ambiente
`source`	Ejecuta los comandos escritos en un archivo. Puede ser utilizado para actualizar el ambiente de la concha
`time`	Muestra el tiempo de ejecución de un comando
`unalias`	Elimina una definición de un alias
`unset`	Elimina una variable de la concha
`unsetenv`	Elimina una variable de ambiente

Tabla 3.1: Comandos de las conchas tipos Bash

El segundo tipo de comando está conformado por una serie de programas cuyo comportamiento es independiente del tipo de concha, esto hace que se incremente la flexibilidad del sistema operativo, pues cada comando es un programa independiente con opciones y modificaciones. Estos comandos se explican en las siguientes secciones.

Para obtener información acerca del uso de estos comandos, GNU Linux cuenta con un manual en línea, que puede ser consultado a través del comando **man** como se muestra a continuación:

man [-k] [comando—palabra clave]

Por ejemplo:

man cp

```
CP(1)                                                          CP(1)

NAME
     cp, ln, mv - copy, link or move files

SYNOPSIS
```

```
     cp [ -firRp ] file1 [file2 ...] target
     ln [ -sif ] file1 [file2 ...] target
     mv [ -if ] file1 [file2 ...] target

DESCRIPTION

     file1 is copied (linked, moved) to target.  Under no
     circumstance can file1 and target be the same (take care
     when using sh(1) metacharacters). If target is a directory,
     then one or more files are copied (linked, moved) to that
     directory.  If target is an existing file, its contents are
     destroyed, except in the ln and ln -s case where the command
     will fail and ln will write a diagnostic message to standard
     error (use the -i or -f option to override this behavior).
     NOTE that this is a change from the historical ln execution.
     ....
```

3.1.4. Gestión de archivos

Listado de archivos

Para mostrar el listado de los archivos del directorio actual se utiliza el comando **ls**, el cual tiene la siguiente sintaxis:

ls [-RadLCxmlnogrtucpFbqisf1AM] [nombres]

Las opciones en el primer corchete pueden ser utilizadas solas o una combinación de ellas. Algunas de las opciones mas utilizadas son:

- **ls**
  Muestra una lista de los archivos del directorio.
- **ls -a**
  Despliega una lista de los archivos pero además muestra los archivos de configuración que suelen llamarse "archivos escondidos" cuyos nombres comienzan con el caracter punto (.).
- **ls -l**
  Despliega una lista detallada de los archivos y directorios. Muestra los permisos, el número de enlaces, propietario, tamaño en bytes y cuando ocurrió la última modificación para cada uno de los archivos.
- **ls -F**
  Muestra una lista de archivos agregando una diagonal (/) al final de los nombres de directorio; un asterisco (*) si se trata de un archivo ejecutable; un arroba (@) si el archivo es un enlace simbolico y un igual (=) si el archivo es un socket.

Ejemplo:

% **ls**

```
Miscellaneous  default.gif    hec.html.bak   index.html     index.shtml
cgi-bin        hec.html       hec01.html     index.html.N
```

% **ls -a**

```
.              cgi-bin        hec.html.bak   index.html.N
..             default.gif    hec01.html     index.shtml
Miscellaneous  hec.html       index.html
```

% **ls -l**

```
total 64
drwxr-sr-x   2 hector   ciencias     512 Apr 19 17:50 Miscellaneous
drwxr-sr-x   2 hector   ciencias     512 Apr 19 17:50 cgi-bin
-rw-r--r--   1 hector   ciencias    2085 Dec 12 1996  default.gif
-rw-------   1 hector   ciencias    2645 Feb 27 1997  hec.html
-rw-------   1 hector   ciencias    1787 Feb 27 1997  hec.html.bak
-rw-------   1 hector   ciencias    1368 Feb 27 1997  hec01.html
-rw-r--r--   1 hector   ciencias     860 Dec 12 1996  index.html
-rw-r--r--   1 hector   ciencias     798 Oct  9 1997  index.html.N
lrwxrwxrwx   1 root     ciencias      10 Apr 25 15:44 index.shtml -> index.html
```

Copiando Archivos

El comando para copiar archivos o directorios tiene la siguiente sintaxis:

- **cp archivo1 archivo2** Copia el contenido del archivo archivo1 en el archivo archivo2.
- **cp arch1 arch2 .... dir1** Cada archivo de la lista será copiado en el directorio dir1. El directorio dir1 debe estar creado con anterioridad.
- **cp -r dir1 dir2** Copia todo lo que esté contenido en el directorio dir1 al directorio dir2. Si dir2 no existe,`cp` lo creará.
- **cp -i archivo1 destino** Si la opción `-i` es especificada, el `cp` preguntará si sobreescribe "destino" en caso de que este ya exista.
- **cp -f archivo1 destino** La opción `-f` especifica que se debe sobreescribir todo sin preguntar.

Ejemplo:

% **cp .cshrc ejemplo1**

Moviendo archivos

mv [-if] file1 [file2 ...] destino

Este comando moverá el contenido del archivo file1 a "destino". si "destino" es un directorio, `mv` lo copiará dentro con el mismo nombre que tenía en su ubicación original. Si "destino" está en el mismo directorio que file1 `mv` funciona cambiando el nombre. Las opciones `-i` y `-f` funcionan igual que en `cp`

Borrando Archivos

rm [-f] [-i] archivo ...

Este comando borrará el o los archivos especificados. Las opciones `-i` y `-f` funcionan igual que en `cp`. Por omisión este comando no pide confirmación y la información eliminada por esta vía no es recuperable, por lo que se recomienda que al trabajar con información delicada se utilice la opción `-i`.

Ejemplo:

% **rm ejemplo1**

Otra forma del comando `rm` es:

rm -r dir1 ...

En este caso el `rm` borra todo el contenido del directorio dir1, incluyendo subdirectorios y archivos ocultos (que empiezan por .).

Visualizando el contenido de los archivos

GNU Linux presenta una serie de comandos que permiten ver el contenido de los archivos de distintas maneras. La forma más básica de desplegar un archivo es con el comando `cat`, el cual tiene la siguiente sintaxis:

- **cat archivo1** Muestra el contenido del archivo archivo1.
- **cat arch1 arch2 > arch3** Concatena o pega los contenidos de los archivos arch1 y arch2 en el archivo arch3.

Ejemplo:

% **cat prot_alinea**

. .

```
>human
VLSPADKTNV KAAWGKVGAH AGEYGAEALE RMFLSFPTTK TYFPHFDLSH GSAQVKGHGK
KVADALTNAV AHVDDMPNAL SALSDLHAHK LRVDPVNFKL LSHCLLVTLA AHLPAEFTPA
VHASLDKFLA SVSTVLTSKY RLTPEEKSAV TALWGKVNVD EVGGEALGRL LVVYPWTQRF
FESFGDLSTP DAVMGNPKVK AHGKKVLGAF SDGLAHLDNL KGTFATLSEL HCDKLHVDPE
NFRLLGNVLV CVLAHHFGKE FTPPVQAAYQ KVVAGVANAL AHKYH
```

```
>goat-cow
VLSAADKSNV KAAWGKVGGN AGAYGAEALE RMFLSFPTTK TYFPHFDLSH GSAQVKGHGE
KVAAALTKAV GHLDDLPGTL SDLSDLHAHK LRVDPVNFKL LSHSLLVTLA CHLPNDFTPA
VHASLDKFLA NVSTVLTSKY RLTAEEKAAV TAFWGKVKVD EVGGEALGRL LVVYPWTQRF
FESFGDLSTA DAVMNNPKVK AHGKKVLDSF SNGMKHLDDL KGTFAALSEL HCDKLHVDPE
NFKLLGNVLV VVLARNFGKE FTPVLQADFQ KVVAGVANAL AHRYH
%
```

Para desplegar un archivo por páginas (pantallas) se utiliza el comando `more`:
more archivo1 Muestra el contenido del archivo archivo1, una pantalla por vez.

Al ejecutar este comando, la máquina mostrará la primera pantalla del contenido del archivo y se detendrá esperando la interacción de usuario. Para mostrar el resto del contenido se puede utilizar la tecla `enter` que permite avanzar una línea a la vez o la barra espaciadora, que permite avanzar por páginas.

También se puede desplegar sólo el final o el principio de un archivo con los comandos `tail` y `head`. Por omisión, estos comandos muestran las 10 últimas líneas y las 10 primeras líneas del archivo, respectivamente

Ejemplo:

% tail ContenidoLinuxBasico

```
        Estados de los procesos
        Como cancelar un proceso


7. Programacion en las conchas
        Lectura y Escritura
        Estructuras de Decision
        Estructuras de Repeticion


%
```

% head ContenidoLinuxBasico

```
TALLER GNU Linux
Contenido

1. Sistema Operativo GNU Linux.
        Historia
        Descripcion
        Caracteristicas
        Componentes



%
```

Un comando bastante útil es el `wc`, ya que cuenta el número de líneas, el número de palabras y el número de caracteres que contiene un archivo.

Ejemplo:

wc prot_alinea

```
          13            60           648 prot_alinea
```

Esta salida quiere decir que el archivo prot_alinea tiene 13 líneas, 60 palabras y 648 caracteres.

También es importante el comando `file` el cual despliega de forma explícita el tipo del archivo que se le pasa como argumento.

Ejemplo:

% **file ContenidoLinuxBasico**

```
ContenidoLinuxBasico: International language text
%
```

Manejo de directorios

Visualizando el directorio de trabajo

pwd Muestra el directorio de trabajo.

Ejemplo:

% **pwd**

```
/home/ciencias/hector/public_html
```

Cambiando el directorio de trabajo

Para cambiarse a un directoiro se utiliza el comando `cd`

- **cd nombredir** Permite cambiarse al directorio nombredir.
- **cd** Permite cambiarse al Directorio Hogar.
- **cd ..** Permite cambiarse al directorio superior.

Ejemplo:

% **cd Fortran**

Creando nuevos directorios

Para crear un nuevo directorio se utiliza el comando `mkdir`

- **mkdir nombredir** Crea un nuevo directorio con el nombre nombredir.
- **mkdir -p camino1/camino2/nombredir** Crea el directorio nombredir en el camino especificado, si uno o varios de los directorios especificados en la ruta no existe, serán creados.

Eliminando directorios

Para borrar un directorio existen dos opciones:

- **rmdir nombredir** Elimina el directorio con el nombre nombredir, sólo si está vacío
- **rm -r nombredir** Borra el directorio nombredir, sin importar si esta vacío o no y además sin preguntar si el usuario está seguro de hacer esto o no.

Ejemplo:
% **rmdir secuencias**

3.1.5. Búsqueda de archivos

Para buscar archivos dentro de un árbol de directorios se utiliza el comando `find`. Dentro de las sintaxis más utilizadas están:

- **find dir -name arch -print** Busca recursivamente a partir del directorio dir el archivo arch, si lo encuentra, muestra el camino donde esta ubicado este archivo.
- **find dir -name arch -exec cmd \;**

Ejemplo:

% **find / -name ContenidoLinuxBasico -print**

```
/gil/latex/Linux/ContenidoLinuxBasico
%
```

% **find -name core -exec rm \;**

```
%
```

3.1.6. Manejo de permisos

GNU Linux proporciona cuentas para múltiples usuarios, asignando a cada cuenta un directorio hogar. Como se indicó en secciones anteriores, cada cuenta le es asignado un identificador numérico y un nombre (login) con los cuales obtiene acceso a la información ubicada en el directorio hogar.

El esquema de seguridad de los archivos está estructurado en tres clases de usuarios. El **dueño**(u) del archivo, el **grupo** (g) al que pertenece el dueño, y los **otros**(o) usuarios que no son el dueño o no pertenecen a su grupo. (La letra "a" se utiliza para representar a todos los usuarios: dueño, grupo y otros).

Cada archivo en GNU Linux posee un atributo para identificar el dueño y el grupo. Además, posee una serie de bits (9 en total) para definir la permisología de lectura escritura y ejecución del archivo. Estos bits están organizados como se muestra en la figura **??**.

Con ayuda de esta estructura se puede definir, para cada archivo, una combinación de permisos para que los usuarios del sistema tengan el acceso adecuado al archivo.

El comando **ls -l** visualiza el estado actual de los permisos de un archivo. A continuación se muestra un ejemplo:
% **ls -l**

```
-rw-r--r--   1 gilberto cecalc      12601 Nov  5 14:41 PROD3.txt
-rw-r--r--   1 gilberto cecalc      17066 Nov  4 16:05 PROD6.txt
-rw-r--r--   1 gilberto cecalc      14829 Nov  6 11:09 PROD7.txt
-rwx------   1 gilberto cecalc        133 Oct 11 08:19 indice.bbl
-rwx------   1 gilberto cecalc       1017 Oct 11 08:19 indice.blg
-rwx------   1 gilberto cecalc      10757 Nov  8 11:48 indice.log
-rwx------   1 gilberto cecalc     109057 Oct  8 09:21 indice.ps
-rwx------   1 gilberto cecalc      75850 Nov  8 15:06 indice.tex
-rwx------   1 gilberto cecalc       4866 Nov  8 11:48 indice.toc
-rw-r--r--   1 gilberto cecalc       2628 Nov  8 11:44 permisos.gif
-rw-r--r--   1 gilberto cecalc     776253 Nov  8 11:45 permisos.ps
-rwx------   1 gilberto cecalc      28786 Oct 11 16:02 prueba.ps
-rwx------   1 gilberto cecalc     163455 Oct  5 09:24 tallerunix
```

La primera columna de información esta conformada por diez caracteres. El primero es una idenficación del tipo de archivo y el resto corresponde a los permisos organizados de la manera en que se muestra en la figura **??**.

Para modificar los permisos de un archivo se utiliza el comando **chmod** y su sintaxis es como sigue:

chmod permisos archivos

Existen dos nomenclaturas para construir el argumentos "permisos" del comando chmod. La primera de ellas consiste en generar un decimal de tres dígitos a partir de la

transformación de los tres octetos que conforman los bits de permisos. Cada grupo de tres bits representa un número binario en el rango comprendido entre cero y siete. Como base de esta primera forma de construir el argumento de permisos se asume que un uno (1) implica asignar el permido y un cero (0) significa negarlo. Si se toma un octeto cualquiera, el del dueño por ejemplo, y se le asigna permiso de lectura, escritura y se le niega el de ejecución, se tiene el número binario 110, lo cual representa al seis en decimal. El mismo procedimiento se aplica a los otros dos octetos. Asi, se puede obtener el número decimal de tres digitos que se necesita en este caso.

Ejemplo:

Asignar todos los permisos para el dueño y el grupo, y solo lectura para el resto de los usuarios de un archivo particular. En el octeto del dueño se tiene 111 lo que es igual a 7. Para el octeto del grupo se tiene el mismo valor 111, es decir, 7. Por último, en el octeto de los otros tenemos 100, es decir, 4. entonces el comando queda de la siguiente forma:

% **chmod 774 archivo**

La otra manera de construir el argumento de permisos es colocar un conjunto de caracteres que representan los permisos a ser asignados. La forma que tendría el argumento es:

ClaseDeUsuario Acción Permiso

Donde **ClaseDeUsuario** es uno o una combinación de los caracteres de la tabla 3.2, **Acción** es un caracter de la tabla 3.3 y **Permiso** es uno o una combinación de los caracteres de la tabla 3.4.

Tabla 3.2: Clase de usuario

Caracter	**descripción**
u	dueño del archivo
g	grupo del dueño
o	los otros usuarios
a	todos los anteriores

El ejemplo anterior, utilizando esta nomenclatura, queda de la siguiente forma:

% **chmod ug+rwx,o=r archivo**

Tabla 3.3: Acciónes sobre los permisos

Caracter	**descripción**
+	asignar
-	negar
=	sobreescribir (los permisos no especificados se niegan)

Tabla 3.4: Tipo de permiso

Caracter	**descripción**
r	lectura
w	escritura
x	ejecución

3.1.7. Manejo de Medios de Almacenamiento Secundario

Los medios de almacenamiento secundario son mayormente utilizados para la elaboración de respaldos de la información contenida en los sistemas de archivos más importantes. Uno de los primeros medios utilizados fueron las cintas (tapes). GNU Linux cuenta con un comando para manipular ese tipo de dispositivos, el comando `tar` (tape archive).

El comando `tar` puede leer el contenido de una cinta:

% **tar tvf /dev/rmt1 [archivos]**

Copiar hacia una cinta:

% **tar cvf /dev/rmt1 archivos**

Y extraer información de una cinta:

% **tar xvf /dev/rmt1 [archivos]**

Este comando es recursivo y puede trabajar sin problemas sobre árboles completos. También puede ser utilizado sobre cualquier otro medio como por ejemplo discos flexibles.

Otra forma de utilización de este comando es empaquetar (no comprime, aunque las últimas versiones tienen esta capacidad) archivos o directorios completos en un solo archivo al cual se le coloca generalmente la extensión .tar

3.2. Utilizando Ambientes de Ventanas

3.2.1. Gestión de Archivos

Los ambientes de ventanas por lo general cuentan con manejadores de archivos los cuales son aplicaciones gráficas que permiten realizar tareas sobre archivos tales como: copiar, mover, borrar visualizar y renombrar.

En KDE podemos encontrar una de las más versátiles y de fácil uso. Su nombre es **konqueror**

Este manejador de archivos puede trabajar tanto con directorios locales como remotos. También ofrece accesorios avanzados como la barra de herramientas y la pre-visualización de los archivos. Así mismo, este es a su vez un explorador de internet y un visualizador universal de documentos. La forma de ejecutarlo es haciendo click en su icono que se encuentra generalmente en el panel o en el menú de inicio.

Al ejecutar el programa aparecerá la ventana de la figura 3.1

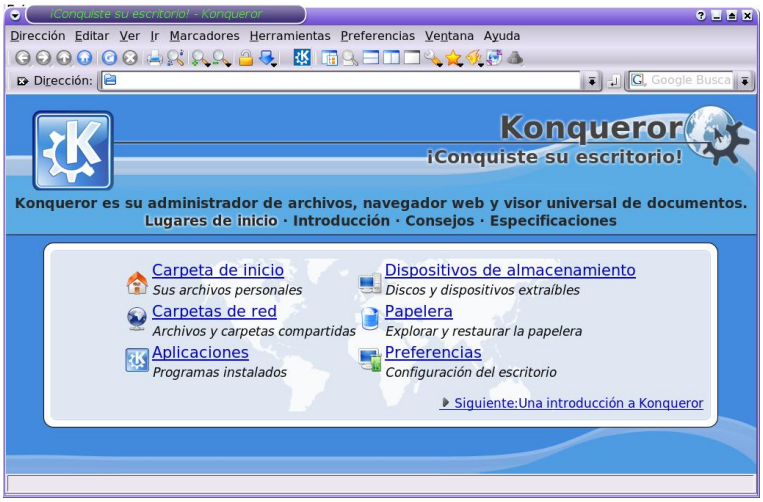

Figura 3.1: Konqueror

- El ícono **Carpeta de inicio** sirve para dirigirse a los directorios locales del usuario.
- **Carpetas de red**, aqui se encuentran las carpetas de la red y las que se comparten.
- **Aplicaciones** haciendo click en el mismo se encontrará todos los programas y aplicaciones instalados.
- **Dispositivos de almacenamiento** se podrá visualizar todos los discos o particiones que hay instalados en la computadora.

- **Papelera** en esta localidad se encuentran todos los archivos que se han borrado y son enviados a la papelera. Cabe destacar que si los archivos aun estan en *Papelera* pueden ser recuperados y utilizados.

3.2.2. Listado de archivos

Utilizando el manejador de archivos *konqueror* esta tarea se realiza por omisión. Al posicionarse en cualquier carpeta o directorio el listado de archivos se presenta de inmediato. Además, la visualización de los archivos se puede realizar de distintas formas: como un listado con el despliegue de las propiedades de los archivos o como íconos. La figura 3.2 muestra el listado de archivos.

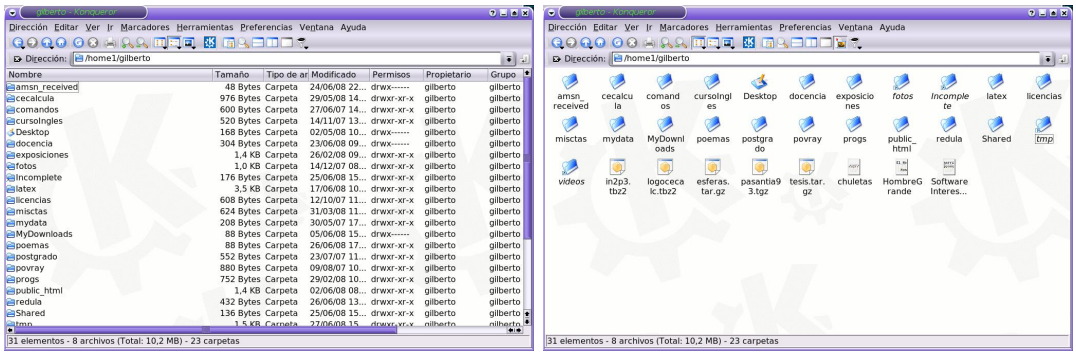

Figura 3.2: Copiando archivos

La forma de listar los archivos se puede seleccionar a través del menú (Ver -¿Modo de vista)

3.2.3. Copiando archivos

El copiado de un archivo se lleva a cabo seleccionando el mismo con el botón derecho del ratón. Inmediatamente se despliega una lista de acciones que se pueden realizar con ese archivo, luego se procederá a hacer click sobre la palabra **"Copiar ctrl + c"**. Véase la figura 3.3

Una vez copiado el archivo, éste se almacena en el portapapeles, luego el usuario debe dirigirse al directorio donde desea copiar el archivo. Ya ubicado en el destino procederá a hacer click el botón derecho del ratón. Igualmente se le despliega una lista en la cual deberá seleccionar la acción **"Pegar ctrl.+v"**. Véase la figura 3.3

Figura 3.3: Copiando archivos

3.3. Moviendo Archivos

Mover un archivo se lleva a cabo seleccionandolo con el botón derecho del ratón, inmediatamente se despliega una lista de acciones que se pueden realizar con ese archivo, se debe proceder a hacer click sobre la palabra *"Cortar Ctrl + X"* como se muestra en la figura 3.3

Luego, sólo es necesario ubicarse en la carpeta que se quiere como destino y hacer click en el botón derecho del ratón y seleccionar *Pegar Ctrl + V*.

3.4. Borrando Archivos

Al momento de eliminar un archivo se debe hacer click con el botón derecho del ratón sobre el archivo, de esta manera se abrirá una lista con distintas opciones, en ella, se debe proceder a hacer click en la opción **"Mover a la papelera"**

Los archivos eliminados se almacenan directamente en una carpeta llamada **trash** (basura) de modo que dichos archivos no se pierden del todo y son recuperables. Si se desea encontrar los archivos eliminados pueden encontrarse en a través del menú **Ir** y luego presionando la opción **sistema**. Se desplegará en la ventana principal el ícono de la papelera. Ver figura 3.4.

Cabe destacar que esto no libera espacio de disco y se debe vaciar la papelera para borrar realmente los archivos, este procedimiento se lleva a cabo de la siguiente forma: El usuario debe seleccionar con un click el icono de papelera ubicado en el escritorio, de esta

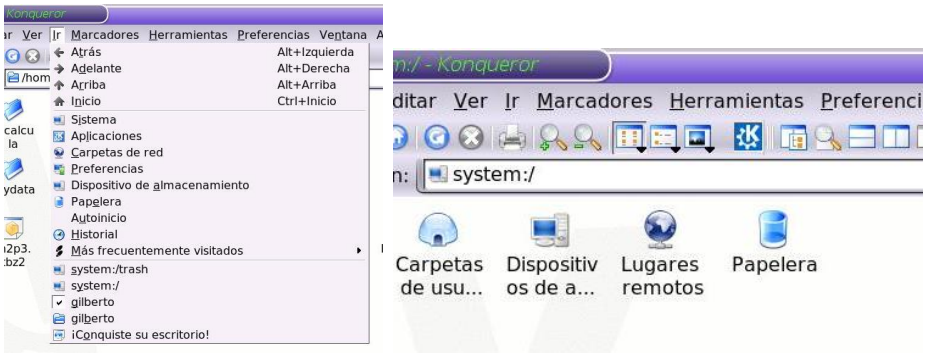

Figura 3.4: Papelera de reciclaje

forma se abrira una lista de opciones en la que se debera seleccionar la opcion "Vaciar la papelera"

3.5. Visualizando el contenido de los archivos

Para visulizar el contenido que esta almacenado en un archivo sólo es necesario hacer click en el archivo y automáticamente se ejecutará una ventana del programa donde ha sido realizado el archivo con la información que el mismo posee, de otra forma también se podra acceder al mismo y ver su contenido, haciendo click sobre el archivo con el botón derecho del ratón y presionando sobre la opción **"Abrir en nueva ventana"** la cual abrirá el mismo en una nueva ventana o, **"Abrir en nueva pestaña"** la cual lo abrirá en la misma ventana pero en una pestaña distinta. (ver figura 3.5)

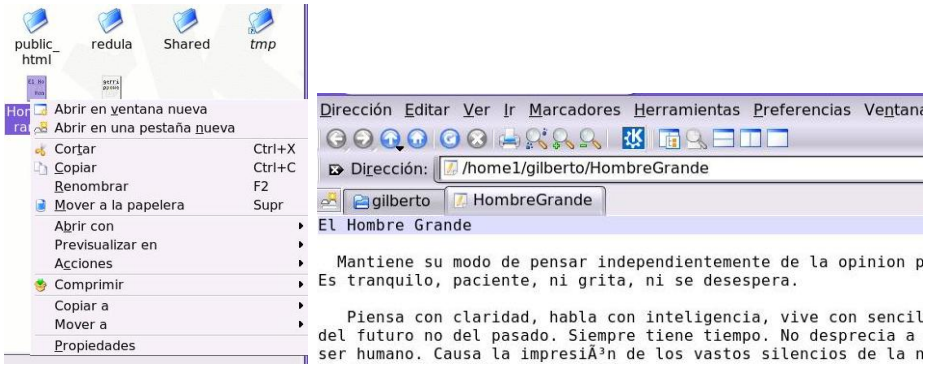

Figura 3.5: Visualizando los Archivos

3.6. Renombrando archivos

Cuando se desea renombrar un archivo, es decir cambiarle el nombre original por otro nuevo, sólo es necesario hacer click con el botón derecho y seleccionar **"Renombrar F2"**. Inmediatamente, el nombre del archivo se hace editable y el usuario puede escribir para colocar el nuevo nombre. Al finalizar, se presiona ENTER y el archivo tendrá un nuevo nombre. Véase la figura 3.6

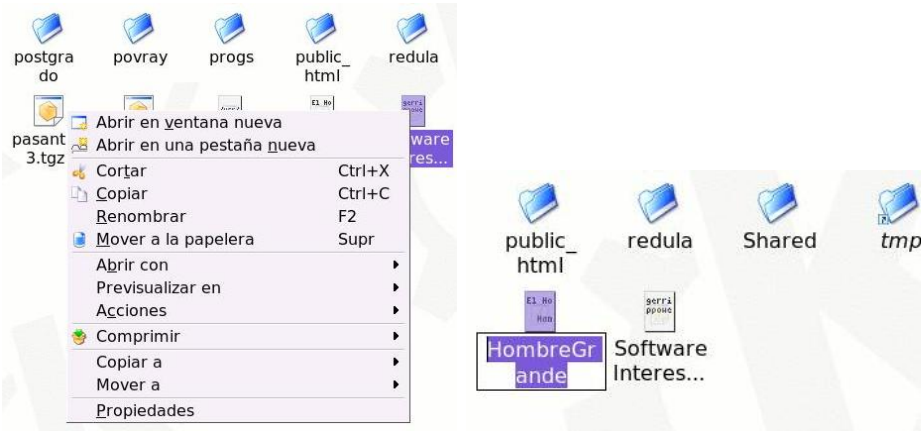

Figura 3.6: Renombrando archivos

Capítulo 4

Edición de Archivos

La edición de textos se realiza en la elaboración de un memo, un correo electrónico, en la modificación o creación de un codigo en C o Fortran, cuando se escribe un reporte o tesis. Por todo esto, la edición de textos es la tarea más común en una computadora. Para tal fin, al igual que otros sistemas operativos, GNU Linux incluye un editor de archivos —el **ed**—, el cual es un editor de líneas poco ágil y engorroso. Por esto, en la mayoría de las distribuciones de GNU Linux se incluye uno o varios editores de texto además del **ed**. El editor de texto más difundido en el ambiente GNU Linux es el **vi**, el cual es un editor que trabaja en ambiente de texto, pero que permite (con la sola interacción del teclado) todas las funciones de los editores de texto mas modernos. Tambien se incluyen algunos editores de texto que funcionan en ambiente gráfico como el xedit, el nedit, etc.

Otro editor que es incluido en muchas distribuciones, sobre todo en el Linux, es el Emacs o su versión gráfica el XEmacs. Este se caracteriza por tener una gran biblioteca de macros que se configuran automáticamente dependiendo del tipo de archivo a editar, permitiendo al usuario resaltar de forma muy sencilla la sintaxis de los lenguajes de programación o de los programas de procesamiento de texto.

En esta sección se hara una introducción al **vi** debido a que es un estándar y no requiere la presencia de un manejador de ventana, lo que permite hacer edición remota de textos, de una forma eficiente. Si bien es cierto que el **vi** puede realizar todas las labores de un editor de textos modernos, su interfaz tipo texto hace un poco laborioso su aprendizaje. Sobre los editores adaptados a los ambientes gráficos sólo se hará un comentario debido a que los ambientes de ventana salen de la cobertura de este curso.

4.0.1. Vi

El **vi** es, hoy en día, el editor de texto por omisión del GNU Linux, y es llamado así por acrónimo de *visually oriented* (la palabra visually se debe a que antes de **vi** sólo existían editores de líneas y teletipos).

El **vi** tiene una serie de características que lo hacen el preferido de las personas que

trabajan a menudo con máquinas conectadas en red, ya sea para edición remota de archivos de entrada o cambios menores en códigos fuentes, como para el trabajo local en edición de archivos más amplios como papers, tesis, etc. Estas características son:

- Procesamiento rápido, especialmente en el arranque y en las operaciones globales. Esto debido a que no tiene el peso de la interfaz gráfica.
- Edición en pantalla completa.
- Modos separados para edición o inserción de texto. Esto se hace nesesario por la imposibilidad de utilizar menus en modo texto, sin embargo hace que la edición de texto sea más segura, ya que no se puede insertar texto mientras se hacen las busquedas y otras operaciones globales.
- Sustitución global y ediciones complejas basadas en comandos `ex`, es cual es la base para toda una familia de editores de archivos en GNU Linux.
- Acceso a comandos del sistema operativo, lo que permite probar la sintaxis de código fuente, entre otros, sin necesidad de salir del **vi**.
- Habilidad de asignar macros a teclas y de personalizar el sistema.
- Posibilidad de efectuar comandos de edición a sectores del archivo, selecionados por número de línea.

Al editar un archivo en **vi** se tienen dos modos: el modo de comandos y el modo de inserción. El modo de comandos es el modo por omisión (al contrario que en la mayoría de los editores de texto para PC.). En este modo se puede navegar por el texto, borrar caracteres, borrar líneas, marcar, mover o borrar bloques, etc. En el modo de comandos se puede tener acceso a los comandos de `ex` comenzando por presionar la tecla ":".

Para cambiarse al modo de inserción se utilizan las letras `i`, `o`, `a`, `I`, `O`, `A`, `cw` y `CW`, dependiendo de la acción que se quiera realizar. Para volver al modo de edición de texto se utiliza la tecla `esc`. Esta tecla también se puede utilizar para cancelar un comando de `ex`.

Los comandos básicos en vi se dividen en varios grupos dependiendo de la función y de la forma de utilización. Los grupos principales son: los comandos de navegación, modificación, comandos de control o subcomandos, búsqueda y de copiado y pegado de texto. Cada grupo se describe brevemente en las tablas 4.1, 4.2, 4.3, 4.4, 4.5, respectivamente.

Tabla 4.1: Comandos de Navegación

comando	descripción
flechas o hjkl	mueve el cursor un caracter.
w	hacia adelante una palabra.
b	hacia atrás una palabra.
e	hacia el final de la palabra actual.
ctrl-u	desplaza el texto hacia abajo media pantalla
ctrl-d	desplaza el texto hacia arriba media pantalla
ctrl-f	desplaza el texto hacia abajo una pantalla
ctrl-b	desplaza el texto hacia arriba una pantalla
0	va al principio de la línea
$	va al final de la línea
:1	va al principio del archivo
:$	va al final del archivo
G	va al final del archivo
n G	a la línea número n
:n	a la línea número n

4.0.2. Otros editores (emacs, pico, joe).

xedit

Es un editor básico que funciona en ambiente X, presenta la ventaja (sobre el vi estándar) de que permite que el usuario utilice el ratón para navegar por el texto, sin embargo tiene muy pocas opciones de edición, de hecho, sólo incluye macros para búsqueda y reemplazo de cadenas de caracteres.

emacs y xemacs

Es el editor de la Fundación de Software Gratis (GNU). Es personalizable, extensible, permite interactuar con el ratón y tiene ayuda en línea. El emacs, al contrario del vi está siempre en modo de inserción y los comandos no hacen diferencia entre mayúsculas y minúsculas. Una buena manera de empezar es por ejemplo:

```
yemanya% emacs [enter]
crtl-h t
```

de este modo se invoca el tutorial, que se puede seguir paso a paso para familiarizarse con la interfaz del emacs.

Tabla 4.2: Comandos de Modificación

comando	**descripción**
i	inserta texto antes del cursor
I	inserta al principio de la línea
a	añade texto después del cursor
A	añade texto al final de la línea
o	abre una nueva línea debajo del cursor
O	abre una nueva línea encima del cursor
x	borra un caracter (delete)
X	borra un caracter (backspace)
dw	borra una palabra
dd	borra una línea
r	reemplaza un caracter
s	sustituye un caracter por una cadena
S	sustituye la línea
cw	sustituye la palabra actual
c# w	sustituye las # palabras consecutivas
C	sustituye el resto de la línea
cc	sustituye la línea
u	deshace el último cambio
U	deshace todos los cambios en la línea actual
J	une líneas
Esc	termina un comando de inserción o reemplazo

Tabla 4.3: Subcomandos

comando	**descripción**
:w	guarda el archivo
ZZ	guarda los cambios y sale del editor
:q!	sale sin guardar los cambios
:e!	edita de nuevo el archivo anterior, es decir, se devuelve a la última version guardada
:n	edita el próximo archivo, si el editor fue invocado con vi arch1 arch2 arch...
:f	muestra el archivo actual
:set nu	enumera las líneas

Tabla 4.4: Comandos de Búsqueda

comando	**descripción**
/texto[enter]	busca el texto "texto" hacia adelante
?texto[enter]	busca el texto "texto" hacia atrás
n	busca la siguiente ocurrencia del texto
N	busca la ocurrencia anterior del texto

Tabla 4.5: Copiar y Pegar

comando	**descripción**
Y o yy	YANK (copia) la línea actual
yw	YANK (copia) la palabra siguiente
p	pega la última copia antes del cursor
P	pega la última copia después del cursor
ma	marca el un extremo de un bloque de líneas
y'a	marca el otro extremo de un bloque de líneas y lo copia

Capítulo 5

Comandos Avanzados

GNU Linux tiene a disposición de los usuarios una serie de herramientas que realizan tareas muy específicas. Además, presenta una característica de modularidad que hace posible combinar esas herramientas y así permitir a los usuarios ejecutar trabajos mucho más complejos.

A continuación se describen algunos de los comandos más útiles.

5.0.3. Ordenamiento de Archivos

sort [-t separador] [-i] archivo ...

El comando `sort` sirve para ordenar el contenido de un archivo. También tiene la capacidad de fusionar diferentes archivos en uno solo, manteniendo cierto orden en los registros.

5.0.4. Búsqueda de Cadenas de Caracteres en Archivos

Para buscar una cadena de caracteres dentro de uno o varios archivos se utiliza el comando `grep`

- **grep cadena arch1** Muestra las líneas del archivo arch1 que contienen la palabra cadena.
- **grep -i cadena arch1** Muestra las líneas del archivo arch1 que contienen la palabra cadena, pero sin distiguir entre mayúsculas y minúsculas.
- **grep -n cadena arch1** Muestra las líneas del archivo arch1 que contienen la palabra cadena, pero añade el número de la línea al principio

Ejemplo:
% **grep slovaca secuencias.genebank**

```
Gb_ba1:Rirrgdx  L36224 Rickettsia slovaca (strain 13-B) 16S ribosomal RNA ..
Gb_ba1:Rsu43808 U43808 Rickettsia slovaca rOmpA (ompA) gene, partial cds.
Gb_ba1:Rsu59725 U59725 Rickettsia slovaca citrate synthase (gltA) gene, p...
Gb_ba2:Rsu83454 U83454 Rickettsia slovaca rOmpA (ompA) gene, partial cds.
%
```

5.0.5. Cortar y Pegar Archivos

Existen comandos para extraer información desde archivos que se encuentren estructurados de forma particular. También en GNU Linux está presente un comando para poder unir información de manera sistematizada proveniente de archivos.

El primero de los comandos es **cut** el cual es capaz de cortar trozos de archivos según un patrón específico.

- **cut -cl1-l2,l3-l4,...,ln-lm archs** Este comando extrae de los archivos archs la información de cada línea comprendida entre los caracteres l1 y l2, l3 y l4 y asi sucesivamente. l1,l2,l3...lm son las posiciones de los caracteres en cada línea.
- **cut -d"sepf1,2,..,n archs** Este comando extrae de los archivos archs las columnas 1,2,..,n las cuales se encuentran separadas por el caracter sep.

Ejemplos:
% **cut -c1-10,20-30 /etc/passwd**

```
root:x:0:0ot:/bin/bas
bin:x:1:1:
daemon:x:2:/sbin:
adm:x:3:4:adm:
lp:x:4:7:lool/lpd:
sync:x:5:0in:/bin/syn
shutdown:xdown:/sbin:
halt:x:7:0in:/sbin/ha
mail:x:8:1ar/spool/ma
news:x:9:1ar/spool/ne
uucp:x:10:var/spool/u
operator:xrator:/root
games:x:12s:/usr/game
gopher:x:1er:/usr/lib
ftp:x:14:5r:/home/ftp
nobody:x:9dy:/:
gdm:x:42:4gdm:/bin/ba
xfs:x:100:t Server:/e
soffice:x:/home1/soff
yasleyda:x::/usr/peop
%
```

% **cut -d":f1,6 /etc/passwd**

```
root:/root
bin:/bin
daemon:/sbin
adm:/var/adm
lp:/var/spool/lpd
sync:/sbin
shutdown:/sbin
halt:/sbin
mail:/var/spool/mail
news:/var/spool/news
uucp:/var/spool/uucp
operator:/root
games:/usr/games
gopher:/usr/lib/gopher-data
ftp:/home/ftp
nobody:/
gdm:/home/gdm
xfs:/etc/X11/fs
soffice:/home1/soffice
yasleyda:/usr/people/yasleyda
%
```

El comando para pegar información proveniente de archivos diferentes es **paste**. Para explicar como funciona este comando supongamos que se tienen dos archivos, cada uno de los cuales contiene una columna de datos. Supongamos que el primero de estos archivos contiene las coordenadas X de cierta ubicación espacial. Ahora supongamos que el segundo archivo contine las coordenas Y y se desea mostrar por pantalla una columna al lado de la otra, entonces debemos ejecutar el comando paste como sigue:

% **paste arch1 arch2**

Donde arch1 y arch2 son los archivos que contienen la información. Este comando contiene otras opciones interesantes, revíselas con el comando man.

5.0.6. Comparación de Archivos

El comando diff se usa para comparar dos archivos de texto. Su función es comparar línea a línea el contenido de los dos archivos y dar como salida aquellos registros que son distintos. La sintaxis general de este comando es como se muestra a continuación:

% **diff arch1 arch2**

También puede usarse el comando sdiff que cumple la misma función que diff pero presenta la diferencia en forma horizontal:

% **sdiff arch1 arch2**

5.0.7. Comparación de Directorios

Este comando permite comparar el contenido de dos directorios y genera información tabulada con el resultado de la comparación. La salida de la comparación que se realiza lista el contenido de cada uno de los directorios comparados, y luego las diferencias entre el contenido de tales subdirectorios. La sintaxis de este comando es como se muestra a continuación:

% **dircmp [-d] arch1 arch2**

La opción **d** muestra el contenido donde difieren los archivos.

5.1. Manejo de Procesos

Ya hemos mencionado la capadidad de GNU Linux para manipular mas de un proceso a la vez. En esta sección se describen las diferentes acciones que se pueden tomar para gestionar los procesos en ejecución dentro de una máquina GNU Linux.

Para comenzar definamos primero el concepto de proceso en el marco del sistema operativo GNU Linux. Un **proceso** es un programa que se ejecuta, y al momento de ser iniciado se genera un descriptor conformado por una estructura de datos que contiene toda la información relacionada con el proceso. Esta estructura puede ser referenciada mediante un número llamado identificador de proceso (Proccess Identifier, PID). El sistema operativo mantiene una tabla con todos los procesos activos en un momento determinado la cual utiliza para la gestión de los mismos.

5.1.1. Estados de los procesos

Los procesos pueden pasar por diferentes estados una vez iniciados. No siempre un proceso se encuentra dentro del procesador sino que puede permanecer en otros estados mientras ocurre algún evento específico o se ejecute alguna operación sobre uno de los dispositivos periféricos del sistema.

En líneas generales los procesos en un sistema operativo multitarea como lo es GNU Linux puede encontrarse en uno de los siguientes estados. Al ser iniciado un programa este es cargado en memoria y es llevado a un estado denominado **listo** donde existe una cola donde competirá con otros procesos por el procesador. Una vez que este es despachado hacia el procesador se dice que el proceso se encuentra en estado de ejecución. El proceso estará dentro del procesador hasta que culmine o hasta que el **quantum** expire para luego regresar al estado de listo. El quantum es un tiempo que se asigna a los procesos para permanecer dentro del procesador. Si el programa se encuentra en ejecución y realiza alguna operación de entrada o salida, entonces el núcleo del sistema lo coloca en un estado **bloqueado**, donde el proceso permanecerá hasta que la operación culmine. Si la operación de entrada/salida tarda demasiado entonces el proceso es llevado a un estado llamado **suspendido-bloqueado** y al proceso se le quita todo recurso que este utilizando. Si la

operación de entrada/salida culmina entonces el proceso se pasa a un estado llamado **suspendido-listo**. En este estado el proceso esta listo para competir de nuevo por el procesador pero no tiene asignado ningún recurso del sistema. Al serle reasignados los recursos al proceso, este pasa de nuevo al estado de listo. Los estados listo, bloqueado y en ejecución son llamados estados activos; el resto son llamados estados inactivos. La figura 5.2 muestra las transiciones de un estado a otro.

Los procesos llamados demonios (daemons) siempre estan listos para cumplir con alguna labor, solo que si ellos permanecieran en estados activos sin hacer nada se estarian desperdiciando los recursos del sistema. Por esta razón ellos se encuentran generalmente en el estado de suspendido-listo o durmiendo.

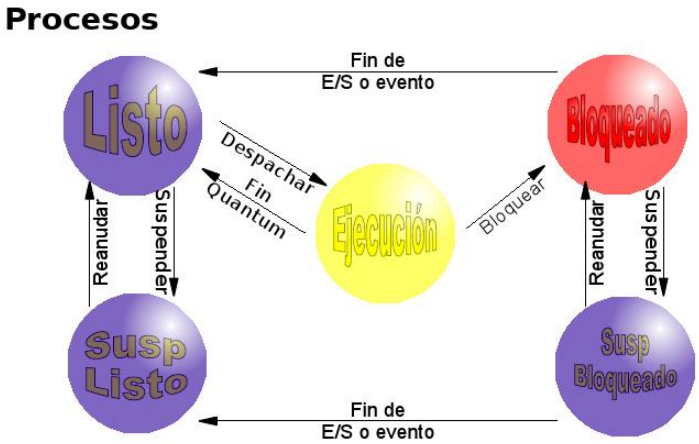

Figura 5.1: Estados de un Proceso

Para observar el estado en que se encuentra todos los procesos del sistema se cuenta con el comando **ps**. La sintaxis de este comando en las versiones System V para desplegar una lista completa de los procesos es:

ps [-edalf]

Ejemplo:

% **ps -edalf**

```
S  UID   PID   PPID C PRI NI ADDR    SZ    STIME    TTY  TIME CMD
A  root     1   0   0  60 20 2805   344   Oct 27     -  1 2:31 init
A  root  2294   1   0  60 20 3046    84   Oct 27     -  1 9:54 syncd 60
A  root  2560   1   0  60 20 d19a   376   Oct 27     -  0:00   errdemon
A  root  3156   1   0  60 20 70ae    56   Oct 27     -  0:00   ssa_daemon
```

...

En la tabla 5.1 se describen algunas de las columnas que son desplegadas cuando se ejecuta el comando ps.

Tabla 5.1: Información de la tabla de procesos

Columna	**símbolo**	**descripción**
PID		Número del proceso.
PPID		Número del proceso padre.
TTY		Terminal vinculado (Los demonios tendrán un ? en este campo).
S		Estado del proceso.
	O	Ejecutándose.
	R	Ejecutable en cola (Running).
	T	Detenido (sTopped).
	D	Esperando en disco.
	S	Durmiente por menos de 20 seg.
	I	Desocupado por más de 20 seg. (Sin el procesador - Inactivo).
	Z	Terminado, control pasa al padre (Zombie).
	X	Esperando más memoria.
TIME		Tiempo de procesamiento.
CMD		Comando que se ejecuta.

5.1.2. Cómo activar un proceso

Algunas versiones de UNIX (SunOS) introdujeron un concepto para describir un comando que se ejecuta: el concepto de tarea o trabajo (job). Un trabajo es un comando

cuya ejecución se ordena desde el terminal. Un trabajo consta de uno o más procesos que se ejecutan en secuencia, bajo la tutela directa o indirecta de una sesión en la concha. Para activar un proceso entonces, la manera más sencilla es invocar su ejecución (que equivale a ejecutar un trabajo) desde la concha del sistema. La invocación consiste en escribir el nombre del archivo que contiene el código ejecutable. Al hacer ésto, la concha entenderá que debe crear un proceso hijo suyo con ese código ejecutable. Más no siempre los procesos son hijos de las conchas o creados en sesiones de usuarios. Existe un conjunto especial de procesos que no dependen de la concha, sino del proceso matríz del sistema (init). Son los llamados demonios del sistema, programas que se ejecutan constantemente y que se emplean comúnmente para atender solicitudes de servicios provenientes de los usuarios u otros programas. Los demonios son activados al encender el sistema, pero pueden reactivarse o cancelarse en cualquier momento. Volviendo con los trabajos, éstos pueden activarse .ªl frente"(foreground)., en cuyo caso la ejecución se "ve.en la pantalla del terminal; ó .ªl fondo"(background) donde el trabajo no despliega ningún mensaje directo a la pantalla. De esta forma, el usuario puede activar varias tareas, mientras que controla cuál de ellas usará la pantalla.

5.1.3. Manipulación de trabajos

Existe una serie de comandos que permiten gestionar los procesos en una máquina GNU Linux. Los shells cuentan con un conjunto de órdenes de control de trabajos que se puede utilizar para mover procesos de modo subordinado (background) a modo principal (foreground). La tabla 5.2 muestra una lista de estos comandos.

Tabla 5.2: Comandos para la gestión de procesos

Comando	**descripción**
CTRL-Z	Suspende el proceso actual
bg	Reanuda el trabajo parado en modo subordinado
fg	Reanuda el trabajo en modo principal
jobs	Lista todos los trabajos parados y todos los trabajos en modo subordinado.
stop	Para la ejecución del trabajo.
&	Coloca el trabajo en modo subordinado cuando este se inicia agregando este símbolo al final de la línea de comandos.

5.1.4. Cómo cancelar un proceso

El núcleo del sistema operativo manipula los procesos a través del envio de señales. Las señales son mecanismos de comunicación interprocesos. GNU Linux cuenta con una serie de llamadas al sistema dedicadas al manejo de señales, pero existe un comando, **kill**, que constituye una herramienta dirigida al usuario no programador, que le permite el envio de señales a los diferentes procesos de los cuales él es dueño. Las señales más comunes se muestran en la tabla 5.3.

Tabla 5.3: Señales más comunes

Señal	**Nombre**	**Descripción**
1	HUP	Reinicia el proceso.
2	INT	Interrumpe el proceso.
9	KILL	Elimina el proceso.
15	TERM	Terminación normal del proceso.

En la jerga de UNIX un usuario propietario de un proceso puede cancelar su ejecución "matando.[el] proceso. Quizas por ello, el comando que permite la eliminación de los procesos se le llama kill. Su sintaxis general es la siguiente:

kill [señal] PID

Ejemplo:

% **kill -9 345**

%

Capítulo 6

Programación en las conchas

Todos los shells de GNU Linux proporcionan un lenguaje de programación interpretado lo cual proporciona una herramienta muy importante que permite combinar comandos para ejecutar trabajos complejos. Esta sección pretende introducir al lector en los detalles de la programación shell de una forma resumida.

En líneas generales todo lo necesario para aprender un lenguaje es tener en cuenta los siguientes puntos: cómo se manejan las variables, cómo leer, cómo escribir, manejo de decisiones y manejo de lazos. La mayoria de los lenguajes presentan estas funcionalidades aparte de otras capacidades propias que puedan tener. En lo que sigue se describen brevemente los puntos anteriores.

6.0.5. Estructura de un script

Cualquier programa shell puede ser introducido directamente sobre la línea de comandos. Cada vez que se introduzca una línea de código, esta será interpretada y ejecutada inmediatamente. Por comodidad, se puede escribir el código en un archivo texto, darle permiso de ejecución y luego correrlo como cualquier otro comando. A esos archivos texto se les llama **scripts**.

Por la presencia de diversas conchas y sintaxis de programación diferentes, se hace necesario distinguir dentro de los scripts el tipo de concha que debe utilizarse para correr un script. Para tal fin, la primera linea del programa indica cual es el tipo de shell. La sintaxis de esta línea es la siguiente:

#!/bin/concha

Donde los posibles valores de "concha" pueden ser: sh, csh, ksh, bash, tcsh, zsh. Lo cual distingue cual será la concha utilizada para interpretar los comandos del script.

Después de esta linea lo que sigue son las intrucciones del programa La mayoria de las veces estas intrucciones están conformadas por los comandos de GNU Linux, lo cual brinda

la oportunidad de ejecutar comandos por lotes.

La diferencia entre la programación de las distintas conchas se basa en la sintaxis de la manipulación de variables, las estructuras de decisión y las estructuras de repetición. En este curso solo mostraremos la programación en Bourne Shell, la cual sirve también para las conchas Korn y Bash.

6.0.6. Manipulación de variables

En la programación shell las variables no poseen tipo y no es necesario la declaración de estas. Para asignar cualquier valor a una variable basta con ejecutar una instrucción como la que sigue:

VARIABLE=valor

Por convención el nombre de las variables en shell siempre es colocado en mayúsculas. Para acceder al valor de una variable se debe hacer referencia a esta de la siguiente forma:

$VARIABLE

Las variables en shell pueden comportarse como listas de valores. La asignación se hace del modo siguiente:

VARIABLE=“valor$_1$ valor$_2$ ... valor$_n$”

Es posible almacenar la salida de un comando en una variable. Esto se hace de la siguiente manera:

VARIABLE=‘comando‘

6.0.7. Lectura y Escritura

Para leer datos desde el teclado y colocarlos como contenido de una variable se utiliza un conjunto de caracteres especiales ($<$), su sintaxis es como sigue:

read VARIABLE

La escritura por pantalla de cualquier texto se realiza con la ayuda del comando echo.

echo comentario

Las conchas GNU Linux utilizan un conjunto de caracteres especiales para realizar funciones como secuenciamiento, encauzamiento, comodines, etc. Estos caracteres son llamados **metacarateres**. Si se desea imprimir algún metacaracter entonces hay que utilizar caracteres especiales que convierten los metacaracteres en caracteres ordinarios. La tabla 6.1 muestra una lista de los metacaracteres y a su vez los caracteres especiales que los convierten en ordinarios.

Tabla 6.1: Metacaracteres

Caracter especial	**Descripción**
’	Elimina el significado especial de ’ < > # * ? & | ; ( ) [ ] ^ , espacios en blancos, tabs.
”	Igual que ’ excepto para $ ‘ ”\
\	Elimina el significado especial del caracter que lo siga.

6.0.8. Manipulación de parámetros en los programas

Ya hemos visto como referenciar el valor de una variable a través del símbolo $. Ahora veremos como manipular cada uno de los parámetros usados al invocar un programa shell: $1,$2, ...$9. Cada uno de ellos permite referenciar a los parámetros 1,2 ...9 respectivamente. El símbolo $* hace referencia a la lista de parámetros completa. Con $# se puede obtener el número de parámetros que conforman una lista o línea de comandos. Esta información puede ser útil cuando se escribe un programa shell que requiera un número exacto de parámetros.

6.0.9. Estructuras de Decisión

En Bash se puede utilizar estructuras de decisión como sigue:

```
if [ decision ]
then
   comandos
else
   comandos
fi
```

También está disponible una estructura de decisión mútiple:

```
case $VAR
in
   valor1)
      comando1
      comando2;;
     .
     .
     .
   valorn)
      comando1
      comando2;;
   *)
      comando1
      comando2;;
esac
```

La tabla 6.2 muestra los caracteres de comparación lógica que pueden ser utilizados dentro de la decisión de las sentencias de decisión y los lazos de repetición que se verán más adelante.

Tabla 6.2: Caracteres de comparación lógica

Símbolo	**Descripción**
-eq	igual que
-ne	diferente que
-gt	mayor que
-lt	menor que
-ge	mayor igual
-le	menor igual
-a	Y lógico
-o	O lógico

6.0.10. Estructuras de Repetición

Lazo while

Una de las estructuras de repetición presentes en la programación Bash es el lazo **while**

```
while [ Condicion ]
```

```
do
   comandos
done
```

Ejemplo:

```
#!/bin/sh

CONT=1

while [ $CONT -le 1000 ]
do
   echo $CONT
   CONT=‘echo $CONT + 1 | bc‘
done
```

Lazo for

El lazo **for** en shell tiene una forma diferente de trabajar, pero le da mayor versatilidad . Esta sentencia trabaja sobre listas, y en cada iteración la variable de control o contador conteniene un elemento de la lista. Ejemplo

```
for i in elemento1 elemento2 elemento3 .... elementoN
do
   comandos
done
```

En la primera iteración la variable de control “i” tendrá como valor el elemento1, en la segunda iteración tendrá elemento2, y así sucesivamente hasta elementoN.

La lista puede estar conformada por la salida de un comando, Por ejemplo:

```
for i in ‘cmd‘
do
   comandos
done
```

En este caso la variable “i” tendrá en cada iteración un elemento de la salida del comando “cmd”.

Ejemplo:

```
for i in `ls`
do
   echo $i
done
```

Capítulo 7

Instalación, Arranque y Parada de una máquina Unix

7.0.11. Instalación

Las plataformas de hardware más comunes donde se puede conseguir el sistema operativo Unix son: estaciones de trabajos RISC, mainframes, supercomputadores, etc. La instalación de estos sistemas varia según la máquina seleccionada y el proceso de instalación presenta grandes diferencias entre una plataforma y otra. Generalmente, cada compañia proporciona un método de instalación bastante sencillo pero se debe tener conocimientos amplios del hardware utilizado, asi como también detalles sobre sistema de archivos, núcleo, etc.

Las nuevas distribuciones de sistemas Unix para PC's proporcionan instalaciones más sencillas y tratan de minimizar la exigencia de experiencia por parte de los usuarios. Los procedimientos de instalación muestran además, gran semejansa entre una distribución y otra.

7.0.12. Arranque (Bootstrapping)

En el momento de inicio de una máquina Unix el núcleo es cargado en la memoria y comienza a ejecutarse, una serie de tareas específicas son ejecutadas antes de estar disponible para los usuarios. [4]

Un proceso típico de Bootstrapping contiene los siguientes pasos:

- Carga e inicio del kernel
- Detección y configuración de dispositivos
- Creación de procesos del sistema:
    - Manejadores de memoria

- Proceso **init**. Este programa utiliza un archivo de configuración (/etc/inittab). Cada linea del archivo tiene la siguiente estructura:

  id:nivel_de_arranque:acción:proceso

  Donde *id* es un identificador de la entrada en el archivo, *nivel_de_arranque* indica en que nivel o niveles debe ser aplicada la entrada. El componente *acción* indica si el proceso **init** debe ejecutar el *proceso* una sola vez (once), ejecutar el *proceso* cada vez que este muere (respawn), esperar hasta que el *proceso* finalice, o deberia ignorar la entrada (off).

- Intervención del superusuario (modo mantenimiento)
- Ejecución de scripts de arranque
- Operación multiusuario

Si un sistema no arranca de formal normal puede deberse a los siguientes problemas:

- Problemas de hardware
- Sectores de arranque defectuosos
- Sistemas de archivos dañados
- Núcleo configurado incorrectamente
- Errores en los scripts de arranque

7.0.13. Parada y Reinicio

En los sistemas operativos comunes para PC, reiniciar, puede ser el tratamiento apropiado para casi cualquier problema. En los sistemas Unix es mejor pensar antes de reiniciar. Las situaciones más comunes para reiniciar es cuando se agrega nuevo hardware, si se modifica un archivo de configuración que se utiliza sólo en el arranque y obviamente, cuando el sistema no responde a ninguno de los procedimientos para restablecerlo.[4]

A diferencia del proceso de inicio, apagar o reiniciar un sistema Unix puede hacerse de varias formas:

- Apagar el interruptor de corriente
- Utilizar el comando **shutdown**
- Utilizar los comandos **halt** o **reboot**
- Enviar la señal TERM al proceso **init**
- Matar el proceso **init**

Capítulo 8

Configuración del Núcleo

El núcleo (*kernel*) es el programa que se encarga de la gestión de todos los recursos de hardware y software presentes en un computador. Este programa constituye realmente el sistema operativo, cualquier otra aplicación es sólo un complemento para interactuar con él. El núcleo siempre se encuentra en la memoria principal para ejercer su labor. El espacio de memoria que éste ocupa es protegido para para evitar que otros programas generen inestabilidad en el sistema. Este programa se encarga de:

- Gestión de la memoria: el núcleo asigna memoria a otros programas que se encuentran en ejecución
- Gestión del procesador: maneja las colas de los procesos y sus prioridades
- Administración de los periféricos: gestiona el uso de todos los dispositivos periféricos del sistema tales como: dispositivos USB, ratón, teclado, monitor, etc.
- Manejo de medios de almacenamiento: organiza la información en dispositivos como discos duros, unidades de CD, floppies, cintas, pen drives, etc.
- Comunicación con otros sistemas: gestión de interfaces de red, etc.

8.1. Arquitectura

La mayoría de los Unix son monolíticos: cada funcionalidad es integrada dentro de un sólo gran programa el cual se ejecuta en modo *kernel*, es decir, cada función se ejecuta bajo la figura de núcleo. En contraste, los sistemas operativos de micro kernel las funciones se realizan utilizando su propia figura y muy pocas de ellas hacen uso del núcleo: programador de tareas sencillo, mecanismos de comunicación entre procesos, etc.

Académicamente los sistemas operativos de micro kernel son preferidos por su sencillez, sin embargo, el costo del pase de mensajes entre sus capas lo hacen más lentos que los

sistemas monolíticos [1]. Teóricamente los sistemas de micro kernel tienen la ventaja sobre los monolíticos de obligar a los programadores a construir sus códigos de forma modular.

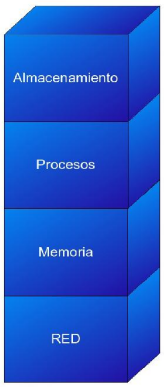

Figura 8.1: Núcleo monolítico

Linux implementa módulos que pueden ser cargados y descargados para cumplir funciones de acuerdo a la demanda de gestión de hardware u otros componentes del sistema.

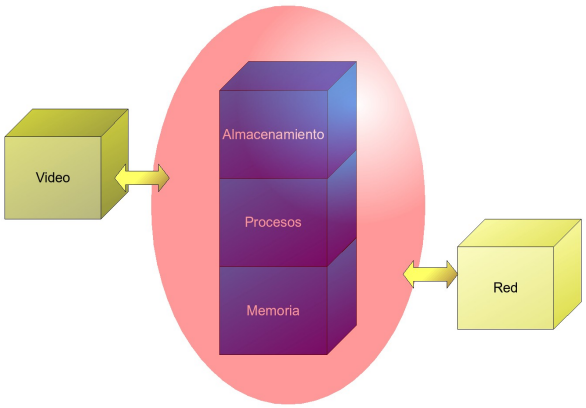

Figura 8.2: Módulos del núcleo

Esta propiedad es interesante pues el tamaño del núcleo se puede mantener bajo así como también el consumo de la memoria.

8.2. Compilación del núcleo

Es deseable mantener un núcleo a la medida que realice sólo las tareas que se necesiten para un determinado sistema. Gracias a que se cuenta con las fuentes del núcleo de Linux es posible añadir o eliminar código y esto se hace a través de pasos sencillos los cuales se describen a continuación:

- Cambiar el directorio de trabajo al lugar donde debe colocarse el kernel

  cd /usr/src/

- Descargar las fuentes del kernel del sitio oficial (www.kernel.org) o de cualquier otro *mirror*

  wget -np -nH http://www.kernel.org/pub/linux/kernel/v2.6/patch-2.6.35.bz2

- Desempaquetar el código fuente

  tar xvjf linux-2.6.35.tar.bz2

- Crear el enlace simbólico a las fuentes del kernel

  ln -snf linux-2.6.35 linux

- Cambiarse al directorio del kernel

  cd linux

- Configurar el kernel, esto es, seleccionar el código que se agregará o se eliminará, utilizando como criterio las funcionalidades del núcleo, y se realiza utilizando distintas interfaces. La más sencilla de ellas es ejecutando el siguiente comando:

  make menuconfig

  Se despliega una ventana como la de la figura 8.3 a través de la cual se puede seleccionar un código determinado para que sea incorporado dentro del kernel como tal (*) o como módulo (M)

- Una vez configurado el núcleo se compila con los siguientes comandos:

  make
  make modules_install

- Copiar el núcleo al directorio de arranque

  cp arch/i386/boot/bzImage /boot/kernel-2.6.23
  cp System.map /boot/System.map-2.6.23

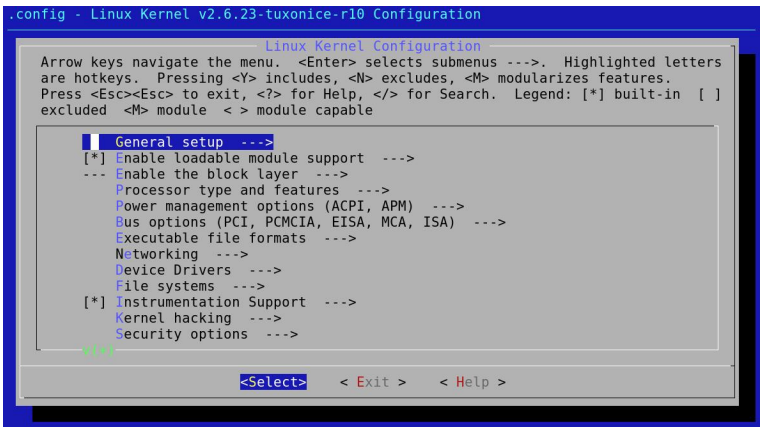

Figura 8.3: Interfaz para la configuración del núcleo

- Editar el archivo /boot/grub/menu.lst para habilitar el nuevo kernel agregando las siguientes líneas

```
# For booting GNU/Linux
title kernel-2.6.23
root (hd0,0)
kernel /boot/kernel-2.6.23 root=/dev/hda1
```

En ciertas ocaciones, y por razones particulares, los usuarios desean configurar ciertas características del núcleo como módulo, pero que son necesarias al momento de inicio de la máquina. Para ello es necesario crear un sistema de archivos temporal en memoria, que se cargue en el momento de arranque, que le provea al kernel los módulos y todas las configuraciones de aplicaciones que requiera para iniciarse de forma correcta. Esto se logra creando una imagen *initrd*, la cual contiene todo lo que el núcleo necesita en el momento de arranque. Esto se hace con el siguiente comando

mkinitrd /boot/initrd-2.6.23 2.6.23

Capítulo 9

Administración de Usuarios

Los procesos y archivos en un sistema Unix tienen asociado el concepto de propiedad. El propietario de un archivo o proceso tiene el control principal sobre éste. Los derechos de propiedad pueden ser superados sólo por el superusuario.

Cada archivo de Unix tiene un propietario y un propietario de grupo. Solamente el propietario puede cambiar los permisos del archivo. Una sóla persona puede ser el propietario de un archivo, mientras que muchas personas pueden ser el propietario de grupo.

El kernel de Unix asocia cuatro números a cada proceso: un UID (User ID) real y efectivo, y un GID (Group ID) real y efectivo. Los números reales son utilizados para la contabilidad y los números efectivos son usados para los permisos de accesos. Estos últimos le dan una imagen diferente a los procesos a la hora de realizar ciertas operaciones sobre archivos con permisos restringidos. Por ejemplo, el comando **passwd** al ser ejecutado por un usuario ordinario necesita cambiar el contenido del archivo /etc/passwd y para hacer esto necesita tener privilegios especiales.

El administrador del sistema necesita tener el control total en un número de situaciones especiales. Para hacer esto posible, Unix trata de forma diferente a una cuanta en particular y es aquella que tiene UID cero (superusuario). Esta cuenta puede realizar las siguientes funciones:

- Montar y desmontar sistemas de archivos
- Cambiar el directorio hogar de un proceso
- Crear archivos especiales
- Fijar el reloj del sistema
- Cambiar la propiedad de archivos
- Fijar prioridades de los procesos
- Fijar el nombre del sistema

- Configurar las interfaces de red
- Detener el sistema

9.0.1. Creación de Usuarios

Para crear y modificar usuarios sólo basta con ejecutar las siguientes tareas:

- Editar el archivo /etc/passwd para definir la cuenta del usuario
- Colocar una palabra clave inicial
- Crear el directorio hogar del usuario
- Agregar el usuario al archivo /etc/group
- Copiar los archivos de ambiente al directorio hogar
- Configurar el correo electrónico del usuario
- Configurar las quotas de espacio en disco

Usualmente sólo las tres primeras tareas son suficientes para crear la cuenta de un usuario y las demás son opcionales y su ejecución depende de la organización que haya planificado el superusuario.

En los sistemas Unix actuales existen herramientas automáticas que facilitan la ejecucion de estas tareas. Unas son del tipo de comandos y otras son más sofisticadas con ventanas gráficas.

El archivo /etc/passwd

El archivo contiene la información principal de los usuarios y cada linea define una cuenta. La estructura de cada linea es:

login:clave:uid:gid:nombre:directorio_hogar:comando

Donde *login* es el nombre con el cual el usuario entra al sistema, *clave* es la contraseña codificada del usuario, *uid* es el identificador del usuario, éste es un número (único dentro del archivo) con el cual es identificado el usuario dentro del sistema, *gid* es el identificador del grupo al cual pertenece la cuenta, *directorio_hogar* es el directorio hogar de la cuenta y por último, *comando*, es el comando que se va a ejecutar cada vez que el usuario entre al sistema.

Ejemplo:

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
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:
news:x:9:13:news:/var/spool/news:
uucp:x:10:14:uucp:/var/spool/uucp:
operator:x:11:0:operator:/root:
```

El archivo /etc/group

Este archivo contiene la información de los grupos del sistema. Cada linea del archivo define un grupo y su estructura es:

nombre_grupo:clave:gid:lista_usuarios

Donde *nombre_grupo* es el nombre del grupo, *clave* es la contraseña, si este campo está vacio entonces no se utiliza, *gid* es el número identificador del grupo y *lista_usuarios* es la lista de usuarios que pertencen al grupo.

```
root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
adm:x:4:root,adm,daemon
tty:x:5:
disk:x:6:root
lp:x:7:daemon,lp
mem:x:8:
kmem:x:9:
wheel:x:10:root
mail:x:12:mail
news:x:13:news
uucp:x:14:uucp
man:x:15:
```

Capítulo 10

Administración de Sistemas de Archivos

Un **Sistema de archivos** es el conjunto de estructuras de datos y algoritmos utilizados para organizar la información. Desde el punto de vista de Unix, un sistema de archivos es una abstracción utilizada por el kernel para organizar los recursos de almacenamiento: discos duros, unidades de CD y floppy. El kernel integra todos estos recursos dentro de una jerarquia de directorios que comienza con el directorio /. Sistema de archivo también se utiliza para hacer referencia a las particiones de un disco.

En un sistema operativo GNU/Linux los archivos se gestionan a través de una estructura de datos llamada **inodo** es utilizada para manejar los archivos. La tabla de inodos es generada en el momento en que un sistema de archivos es creado. Cada inodo contine acerca de cuarenta datos pero dentro de los más importantes podemos mencionar: dueño, grupo, permisos, tamaño, última modificación, último acceso y tipo. Los diferentes tipos de archivos que se pueden encontrar en un sistema operativo Unix son: archivos ordinarios, directorios, archivos de dispositivos por caracteres, archivos de dispositivos por bloques, sockets de dominio, pipes nombrados, enlaces fuertes y enlaces simbólicos.

10.1. Montando un sistema de archivos

Para hacer disponible (montar) un sistema de archivos y poderlo ver como un directorio más del árbol principal se debe ejecutar el comando **mount**. Ejemplo:

```
# mount /dev/hda1 /mnt
```

El argumento */dev/hda1* es el dispositivo correspondiente a la partición donde se encuentra el sistema de archivos y el argumento */mnt* es un directorio vacio desde donde se podrá acceder al mismo.

El comando **umount** es usado para desmontar el sistema de archivos.

 # umount /mnt

Existen comandos para gestionar algunas tareas sobre los sistemas de archivos. La tabla 10.1 muestran algunos de estos comandos.

Tabla 10.1: Comandos de administración de archivos

comando	**descripción**
df	muestra información referente a los tamaños, espacio ocupado y espacio libre de los sistemas de archivos.
du	despliega el espacio ocupado de un archivo o directorio.
lsof	muestra los archivos actualmente abiertos.

10.1.1. El archivo /etc/fstab

Para que los sistemas de archivos estén disponibles de forma automática cada vez que se reinicie la máquina, se utiliza el archivo fstab para indicar los detalles de montaje de cada sistema de archivos. Cada línea de este archivo corresponde a un sistema de archivos y su sintaxis es:

dispositivo pto_de_montaje tipo opciones fs_freq fs_pass

Donde *dispositivo* es la partición u otro dispositivo, *pto_de_montaje* es el directorio donde será montado el sistema de archivos, *tipo* es el tipo de sistema de archivos, *opciones* son las opciones que le son pasadas al montaje, *fs_freq* no es utilizado generalmente, y por último, *fs_pass* indica si debe realizarse un chequeo automático en el momento de arranque de la máquina. Algunas versiones de Unix han cambiado este esquema y es necesario revisar la ayuda para conocer los detalles. Como un ejemplo, se muestra este archivo de una máquina Linux.

```
/dev/hda2   /            ext2    defaults        1 1
/dev/hda1   /c           vfat    defaults        0 0
/dev/hda4   /d           auto    defaults        0 0
/dev/cdrom  /mnt/cdrom   iso9660 noauto,owner,ro 0 0
/dev/hda3   swap         swap    defaults        0 0
/dev/fd0    /mnt/floppy  ext2    noauto,owner    0 0
none        /proc        proc    defaults        0 0
```

10.1.2. Recuperación de Sistemas de Archivos

Para agilizar las operaciones de escritura y lectura que se realizan en un determinado archivo, Unix utiliza un buffer para mantener una copia temporal de los cambios realizados en dicho archivo. Con cierta frecuencia los cambios son actualizados en el archivo real. Si la máquina se apaga de forma anormal, puede que algunos sistemas de archivos queden marcados como sucios, es decir, contienen inconsistencias. Para solucionar esto y llevar al sistema de archivos dañado al último estado consistente, se utiliza el comando:

 # fsck -y *dispositivo*

Donde *dispositivo* es el archivo especial que maneja el sistema de archivos. Este comando debe ser ejecutado cuando el sistema de archivos no está montado. La opción -y obliga al comando a recuperar cualquier error o inconsistencia.

10.1.3. Gestión de Espacio de Disco

Es frecuente encontrarse con usuarios que consumen espacio de disco de forma desenfrenada. Mantener controlados a estos usuarios puede ser difícil. Algunos administradores utilizan scripts para realizar estas tareas. Lo más conveniente para solvertar esta situación es utilizar **quotas de disco**. Esto permite limitar el número de inodos y bloques de disco que pueden ser usados por cada usuario. El número de inodos determina cuantos archivos puede un usuario poseer y el número de bloques determina cuanto espacio de disco puede ocupar.

Información sobre la quotas es mantenida en un archivo de nombre **quotas** ubicado en el directorio raiz del sistema de archivos. Este archivo contine los límites de cada usuario y un resumen del espacio consumido por estos. Algunos sistemas de archivos soportan quotas de grupo y en estos podemos encontrar los siguientes archivos de control: **quotas.user** y **quotas.group**

El comando **edquota** edita los límites definidos en los archivos de quota. Los comandos **quota** y **repquota** muestran información sobre los límites y el espacio de disco utilizado.

El comando **quotacheck** examina bloque por bloque para calcular el espacio de disco usado actualmente, después, actualiza el archivo de quotas.

Para habilitar las quotas se debe tener soporte en el kernel. Si no se tiene tal soporte se debe compilar un nuevo kernel. Luego, se debe arrancar las quotas explícitamente ejecutando los siguientes comandos: **quotacheck -a** y **quotaon -a** después de haberse montado todos los sistemas de archivos correctamente.

10.2. Arreglos de discos

La tolerancia a fallas y la alta disponibilidad son características muy deseables cuando manejamos información importante. Linux cuenta con una serie de mecanismos en su núcleo para la gestión de sistemas de archivos con diferentes niveles de redundancia. Dentro de los mecanismos más utilizados podemos encontrar:

- Arreglos de discos redundantes (RAID, Redundant Arrays of Inexpensive Disks)
- Grupos de volúmenes lógicos (LVM, Logical Volume Manager)
- Discos redundantes en red (DRBD)

10.2.1. RAID

Es una estrategia para combinar varios discos "pequeños" independientes para conformar un arreglo que ofrezca mejores prestaciones que uno sóla unidad de gran tamaño. Este arreglo puede ser visto como un sólo disco duro. El método de organización de la información depende del **nivel RAID**. El tiempo medio entre fallas de un arreglo de discos es igual al tiempo medio entre fallas de cada disco dividido entre el número de discos que constituyen el arreglo. Esto puede no satisfacer los requisitos de algunas aplicaciones, sin embargo, se puede configurar discos redundantes para tener un buen nivel de tolerancia a fallas. En esta sección utilizaremos los términos disco duro y dispositivo indistintamente.

Exiten distintos niveles RAID que establecen la organización del arreglo: RAID lineal, RAID 0, RAID 1 RAID 2, RAID 3, RAID 4 y RAID 5. Los niveles 2 y 3 sólo son utilizados en configuraciones muy especializadas y el kernel de Linux no los soporta. Definamos algunos términos antes de describir los detalles de los distintos enfoques:

- S: tamaño de un disco duro
- N: número de discos en un arreglo
- P: rendimiento de un disco medido en MB/seg

RAID lineal

Este enfoque coloca los discos duros uno detrás de otro conformando un sólo gran dispositivo de almacenamiento. Las escrituras siempre van al primer disco, una vez que este se llene se hacen en el siguiente y así susecivamente hasta ocupar el espacio de todos los discos. No se cuenta con redundancia. Como se muestra en la figura 10.2.1 los archivos se van copiando uno detrás del otro llenando el primer disco, luego el siguiente y así sucesivamente.

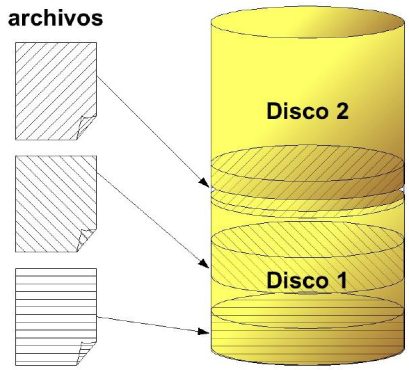

Figura 10.1: RAID lineal

RAID 0

Este nivel es conocido como "rayado" o "stripe". Aquí las operaciones de lectura y escritura se realizan en paralelo en los discos involucrados, por lo tanto, su rendimiento se incrementan siempre. Si el bus de la máquina es lo suficientemente rápido se puede obtener hasta casi $N * P$ MB/seg. Se recomienda que todos los discos tengan el mismo tamaño para que el rendimiento no se vea afectado. Este nivel tampoco cuenta con redundancia. Como se muestra en la figura 10.2 cada archivo es particionado en N porciones y cada porción es colocada en un disco distinto.

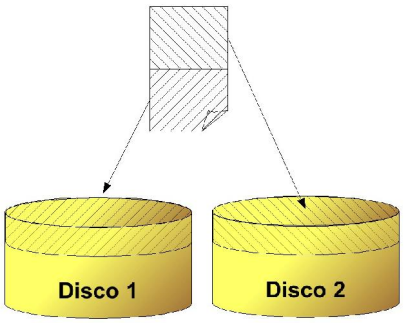

Figura 10.2: RAID 0

RAID 1

Este es el primer enfoque que introduce tolerancia a fallas. Se duplica la información de un disco duro en otro disco duro. Se puede tener más de un disco duro para duplicar la información. En este nivel el rendimiento de las operaciones de lectura se incrementa

hasta casi $N * P$ MB/seg mientras que el de las escrituras es el mismo que para un sólo dispositivo. Es recomendable que cada disco duro que conforma el arreglo se encuentre en un controlador de disco distinto para maximizar el rendimiento. En la figura 10.3 un archivo es escrito al mismo tiempo en cada disco del arreglo. De esta forma se cuenta con respaldo de la información.

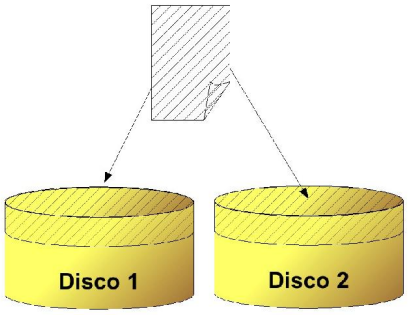

Figura 10.3: RAID 1

RAID 1+0

Para aprovechar obtener el rendimiento proporcionado por el RAID 0 y la tolerancia a fallas del RAID 1 podemos combinar ambos enfoques en lo que se conoce RAID 0+1. En este caso debemos contar con varios discos duros. La figura 10.4 muestra la organización del arreglo. Primero se construyen RAIDs 1 con pares de discos y estos se utilizan para conformar un RAID 0.

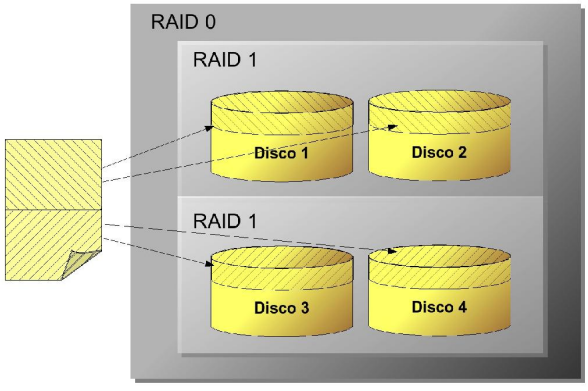

Figura 10.4: RAID 0+1

RAID 4

Este tipo de arreglo utiliza striping a nivel de bloques, es decir, la unidad de gestión de la información es un bloque de datos. Para construir un RAID4 se necesita al menos 3 discos. Uno de los discos duros del arreglo es dedicado para gestionar la paridad; esto permite que cada miembro del arreglo actue de forma independiente cuando un sólo bloque es solicitado. Sin embargo, el disco de paridad puede convertirse en un cuello de botella. Por esto, este tipo de arreglo fue rápidamente remplazado por RAID5.

RAID 5

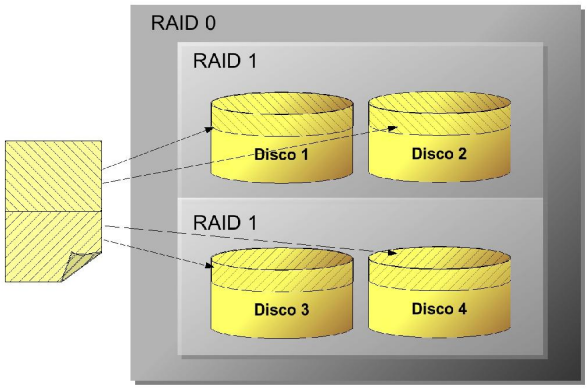

Figura 10.5: RAID 5

10.2.2. Grupos de volúmenes lógicos

El kernel de Linux ha incorporado un gestor de volúmenes lógicos (LVM, Logical Volume Manager) que ofrece algunas ventajas sobre los arreglos de discos anteriores. Este proporciona un nivel más alto de los dispositivos de almacenamiento de un computador que los sistemas tradicionales, dándole mayor flexibilidad al administrador para gestionar el espacio de almacenamiento para aplicaciones y usuarios.

La principal ventaja de LVM sobre otros enfoques es la posibilidad de establecer un espacio mínimo para un sistema de archivos y aumentarlo en caliente [1] a medida que se necesite. Asi mismo, se puede realizar una reconfiguración de los sistemas de archivos sin tener que reparticionar el disco duro.

Otra ventaja importante es que LVM nos brinda la posibilidad de crear sistemas de archivos de mucho más capacidad que la de cualquier disco duro del mercado. Esto se logra

[1] Sin reiniciar o desconectar el sistema

gracias a que LVM agrupa varios dispositivos de almacenamiento y genera una grupo de volúmenes (*Volume Group*) de donde se puede sacar espacio de almacenamiento a medida que se necesite. En la figura 10.5 este está representado por *MIvg*. Los sistemas de archivos de crean sobre volúmenes lógicos (*Logical Volumes*) los cuales son una especie de partición virtual cuyo tamaño se define en el momento de su creación y puede ser cambiado luego de forma fácil, sin que la información se elimine. En el ejemplo de la figura 10.5 los sistemas de archivos /home, /tmp y /opt tienen su información en distintos discos duros. Como se observa en la figura los volúmenes lógicos son excluyentes y también puede quedar espacio sin utilizar.

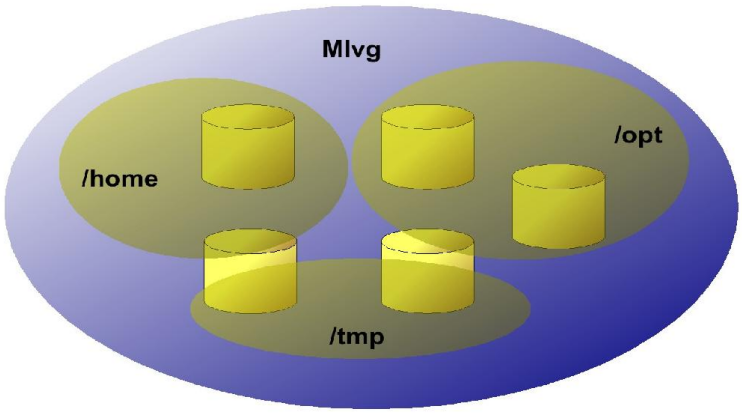

Figura 10.6: Logical Volume Manager

10.2.3. Ejercicio

La siguiente actividad práctica muestra como crear un sistema de almacenamiento con tolerancia a fallas y de altas prestaciones. La redundancia se logrará utilizando RAID 1 con varios pares de discos y el alto rendimiento se hará con *stripped LVM*.

1. Primero debemos asegurarnos de tener el soporte correcto en el kernel de Linux

```
Devices Drivers -->
   Multi-device support (RAID LVM) -->
  [*] Multiple devices driver support (RAID and LVM)
  <*>   RAID support (NEW)
  <*>   Device mapper support (NEW)
```

2. Configuración RAID

- Se instala la aplicación **mdadm**

```
% apt-get install mdadm (Linux Debian)

% emerge mdadm (Linux Gentoo)
```

- Crear los RAIDs (Suponemos que se cuenta con 12 discos SATA)

```
% mdadm --create --verbose /dev/md1 --level=raid1 --raid-devices=2 /dev/sda /dev/sdb

% mdadm --create --verbose /dev/md2 --level=raid1 --raid-devices=2 /dev/sdc /dev/sdd

% mdadm --create --verbose /dev/md3 --level=raid1 --raid-devices=2 /dev/sde /dev/sdf

% mdadm --create --verbose /dev/md4 --level=raid1 --raid-devices=2 /dev/sdg /dev/sdh

% mdadm --create --verbose /dev/md5 --level=raid1 --raid-devices=2 /dev/sdi /dev/sdj

% mdadm --create --verbose /dev/md6 --level=raid1 --raid-devices=2 /dev/sdk /dev/sdl

% mdadm --create --verbose /dev/md0 --level=raid0 --raid-devices=6 /dev/md1 /dev/md2 \
/dev/md3 /dev/md4 /dev/md5 /dev/md6
```

- Se escribe el archivo de configuración con los comandos

```
% echo 'DEVICE /dev/sd[a-l]' > /etc/mdadm/mdadm.conf

% mdadm -Ds |grep  num-devices >> /etc/mdadm/mdadm.conf
```

  El archivo queda parecido a lo que sigue:

```
DEVICE /dev/sd[a-l]
ARRAY /dev/md6 level=raid1 num-devices=2 UUID=efdc1d30:90a6d47e:ec941805:14155e59
ARRAY /dev/md5 level=raid1 num-devices=2 UUID=1d9ec183:93cd4a74:a78d5f8c:4e3de3b1
ARRAY /dev/md4 level=raid1 num-devices=2 UUID=860ff8cd:4a8ebd02:99eafa09:8a371ea0
ARRAY /dev/md3 level=raid1 num-devices=2 UUID=e69a862c:a401dadb:fde8f3c0:4d4eb157
ARRAY /dev/md2 level=raid1 num-devices=2 UUID=5de81e45:33b8eea6:79410720:8c6cf8f7
ARRAY /dev/md1 level=raid1 num-devices=2 UUID=aabe0d22:d1952cef:9c8e8e70:93a009c4
```

- Chequear que están activos los RAIDs 1 con el comando

```
% cat /proc/mdstat
```

- Activar el *mdadm* en el tiempo de reinicio de la máquina

3. Configuración LVM

- Editar el archivo /etc/lvm/lvm.conf y modificar las siguientes lineas para que se vean así:

```
md_component_detection = 1

\# Exclude the cdrom drive
filter = [ "r|/dev/cdrom|", "r|/dev/hd.*|" ]
```

- Crear los volúmenes físicos

```
% pvcreate /dev/md1

% pvcreate /dev/md2

% pvcreate /dev/md3

% pvcreate /dev/md4

% pvcreate /dev/md5

% pvcreate /dev/md6
```

- Crear el grupo Lógico (Volume Group)

```
% vgcreate MIvg /dev/md1 /dev/md2 /dev/md3 /dev/md4 /dev/md5 /dev/md6
```

- Crear el volumen Lógico (Logical Volume)

```
% lvcreate -i6 -I4 -L200G -nhomelv MIvg
```

- Crear el sistema de archivos

```
% mke2fs -b 4096 /dev/MIvg/homelv
```

- Montar el sistema de archivos y configurarlo en el /etc/fstab

```
% /dev/MIvg/homelv       /home/ftp          ext2    noatime  0 1
```

Capítulo 11

Administración de Interfaces de Red

Configurar una interfaz de red es relativamente fácil, para ello se utiliza el comando **ifconfig**. La sintaxis de este comando es como sigue:

 # ifconfig *interfaz dirección* **netmask** *máscara* **broadcast** *broadcast*

Donde *interfaz* es el nombre de la interfaz de red que se está configurando, *dirección* es la dirección IP, *máscara* es la máscara de la red y *broadcast* es la dirección que se utiliza para difundir mensajes a toda la red.

Para desplegar las diferentes interfaces de red que están presentes en una máquina puede ser utilizado el siguiente comando:

 # netstat -i

Este comando también es útil para supervisar una interfaz específica, de manera tal de obtener algunas estadísticas de uso de la interfaz. La sintaxis siguiente se utiliza para tales fines:

 # netstat -i *interfaz*

Donde *interfaz* es la interfaz que se quiere obtener información. Recuerde que algunas opciones de los comandos cambian entre las diferentes versiones de Unix, por esto, se recomienda leer el manual del Unix que esté utilizando.

Una vez configurada la interfaz de red, se debe establecer la ruta mínima para poder establecer comunicación con cualquier otra máquina de la misma red. Esto se hace mediante el comando **route**. La sintaxis es como sigue:

 # route add -net *red* **netmask** *máscara interfaz*

Donde *red* es la dirección de la red, *máscara* es la máscara de la red e *interfaz* es la interfaz de red a través de la cual se llega a la red.

Si la red local está conectada a otras redes a través de un enrutador (*gateway*), entonces es necesario agregar una ruta para que la máquina pueda establecer conexiones con equipos de las redes foráneas. Esta ruta es llamada ruta por omisión (*default route*). El siguiente comando es utilizado (sintaxis Linux) para crear esta ruta:

 # route add -net default gw *ip_router*

Donde *ip_router* es la dirección IP del enrutador local. Para revisar las rutas que ha configurado se utiliza el siguiente comando:

 # netstat -rn

Las rutas hasta ahora agregadas, conforman lo que se llama tabla mínima de enrutamiento. Estas permiten la conexión de una máquina común con el resto de las máquinas de la red local y con máquinas de otras redes. Este curso no trata niveles de enrutamiento más complejos como los que se encuentran en equipos especializados (enrutadores), sólo se pretende mostrar lo necesario para conectar un sistema Unix con otros equipos.

Otro comando útil en la supervisión de las interfaces es el comando **ping**. Este comando envia paquetes de tamaño determinado a una interfaz y esta devuelve un paquete de tamaño similar, desplegando el tiempo del viaje de los dos paquetes. Ejemplo:

```
#ping odie
PING odie (150.185.138.43) from 150.185.138.43 : 56(84) bytes of data.
64 bytes from odie (150.185.138.43): icmp_seq=0 ttl=255 time=88 usec
64 bytes from odie (150.185.138.43): icmp_seq=1 ttl=255 time=100 usec
64 bytes from odie (150.185.138.43): icmp_seq=2 ttl=255 time=95 usec

--- odie.cecalc.ula.ve ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.088/0.094/0.100/0.009 ms
```

Capítulo 12

Procesos Periódicos

En UNIX es posible ejecutar procesos automáticamente a una hora determinada, cuando la carga del sistema sea baja, o periódicamente. Esta posibilidad es muy útil para el administrador, ya que le permite automatizar algunas de sus tareas. Por ejemplo, podría diariamente, en momentos de baja carga, ejecutar un proceso que elimine entradas indeseadas en el archivo .rhosts de los usuarios. Para hacer uso de estas facilidades se utilizan los comandos **at**, **batch** y **cron**.

12.0.4. El Comando at

Este comando es usado para correr un comando una sola vez, en un momento especificado. La sintaxis de este comando es at hora, donde la hora puede especificarse en un casi cualquier formato. Además se puede especificar la fecha en que se desea correr el comando. Al escribir el comando y presionan la tecla enter, se tiene oportunidad de introducir la lista de comandos a ejecutar línea por línea. Se termina la introducción de comandos mediante la secuencia de teclas **CTRL-D**. La salida de los comandos ejecutados es enviada por correo al dueño del proceso. Con este comando, usando la opción **-l**, se puede observar la lista de trabajos pendientes. Usando la opción **-r** se pueden eliminar de la lista.

12.0.5. El Comando batch

Este comando permite ejecutar procesos en momentos en que la carga del procesador sea baja. Los comandos se indican de la misma forma que con el comando at. La salida se le entrega al usuario vía correo electrónico. Con batch, no hay forma de predecir cuando se ejecutarán los comandos. Puede ser incluso posible que se ejecuten inmediatamente, si el sistema está descargado.

12.0.6. El Comando cron

Esta es una facilidad que permite automatizar trabajos y ejecutarlos periódicamente. La ejecución de estos comandos se basa en un horario, conocido como archivo crontab. El archivo crontab está formado por líneas con la siguiente estructura.

minutos horas día_del_mes mes día_de_la_semana comando

Los primeros 5 campos son numéricos, donde se puede indicar un número único, un rango de números, una lista de números o un asterisco para indicar cualquier valor. Mediante combinaciones de estos valores el usuario define cuándo debe el comando ser ejecutado.

Por ejemplo, la siguiente línea le pide a cron la ejecución del comando date cada quince minutos, de lunes a viernes, enviando la salida a la consola.

0,15,30,45 * * * 1-5 /usr/bin/date >> /dev/console

El archivo crontab debe modificarse mediante el uso del comando **crontab -e**. Este comando llamará un editor (usualmente vi), donde el usuario podrá modificar las entradas del horario. El administrador puede controlar el uso de estos comandos, impidiéndoselo a sólo algunos usuarios, permitiéndoselo a usuarios selectos, o negándoselo a todos los usuarios. Esto se hace, para los comandos at y batch, a través de los archivos **/etc/at.deny** y **/etc/at.allow**. Para el comando cron se hace mediante los archivos **/etc/cron.deny** y **/etc/cron.allow**. En caso se existir el archivo /etc/at.deny, se le impide el uso únicamente a los usuarios allí indicados. Si el archivo existe pero está vacío, cualquier usuario lo podrá usar. De existir el archivo /etc/at.allow, éste se leerá antes que el anterior, y únicamente los usuarios allí mencionados tendrán permiso para usar el comando. Si no existe ninguno de los archivos indicados, todos los usuarios podrán hacer uso del comando. La misma explicación reza para los archivos /etc/cron.deny y /etc/cron.allow.

Capítulo 13

Seguridad y Supervisión del Sistema

13.1. Seguridad

Originalmente, Unix fue desarrollado sin pensar en sistemas seguros, pero actualmente se ha trabajado bastante en el área y se ha conseguido niveles de seguridad importantes. Aunque el tema de seguridad en redes es bastante amplio y requiere de un curso completo, esta sección intenta especificar una serie de normas dirigidas a la configuración de un sistema Unix seguro.

13.1.1. Definiciones básicas.

A continuación se presentan algunas definiciones básicas relacionadas con la seguridad en sistemas de computación. Para el lector interesado existe un extenso glosario de términos de seguridad en [3]

- **Seguridad en Computadores:** Es la protección contra riesgos de daño de los computadores y los elementos asociados con éstos: la edificación, los terminales, impresoras, discos, cables y otros, además de la información almacenada en el sistema. Una definición más amplia relaciona la seguridad con tres principios básicos: *Confidencialidad, Integridad y Disponibilidad*.

- **Seguridad Administrativa:** Son las reglas y procedimientos de administración que dan como resultado la protección del sistema de computación y sus datos.

- **Política de Seguridad:** El conjunto de leyes, reglas y prácticas que regulan la forma en que una organización maneja, protege y distribuye la información sensible.

- **Autenticación:** Es el proceso de comprobar que un ente (usuario o sistema) es quien dice ser. Es una medida utilizada para verificar la elegibilidad del ente y la autorización del mismo para tener acceso a cierta información. La autenticación protege

contra el uso fraudulento del sistema o la transmisión fraudulenta de información entre el sistema y el exterior. Hay 3 maneras clásicas de autenticarse:

- Algo que el usuario conoce (por ej. una clave o contraseña).
- Algo que el usuario tiene (por. ej. una tarjeta).
- Algo que el usuario es (por ej. alguna característica fisiológica o de comportamiento).

- **Ataque:** Un intento de burlar los controles de seguridad en un sistema. Un ataque puede o notener éxito, dependiendo de la vulnerabilidad del sistema y de la efectividad de las medidas que se tomen para contrarrestar el ataque.
- **Vulnerabilidad:** Es una debilidad en el sistema de computación o un punto susceptible de ser atacado. Las vulnerabilidades pueden usarse para violar la seguridad del sistema.
- **Sitio:** Es cualquier organización que posee computadores o recursos que se relacionen con acceso a redes. Estos recursos pueden incluir, computadores anfitriones (hosts), enrutadores, terminales, servidores de terminales, Computadoras Personales ú otros dispositivos que tengan acceso a Internet.
- **Internet:** Es el conjunto de redes y máquinas que usan el protocolo TCP/IP, conectados a través de pasarelas y compartiendo espacios de nombres y direcciones comunes [2].

13.1.2. Principios básicos de la seguridad en sistemas de computación:

cuando se habla de seguridad en computación es necesario considerar los siguentes pricipios, pues de ellos dependen las acciones a tomar:

1. **Confidencialidad:** Significa que solo los usuarios autorizados por el propietario de los datos pueden accesar la información.
2. **Integridad:** Se relaciona con la protección de datos y programas contra las modificaciones o la eliminación no autorizada.
3. **Disponibilidad:** Garantía de que el servicio no se degrade o no esté disponible sin autorización.
4. **Consistencia:** Asegurar que el sistema trabaje como es de esperarse y no presente comportamiento indeseado.

5. **Aislamiento:** Significa regular el acceso al sistema de personas o programas.

6. **Auditoría:** Llevar y examinar registros de la actividad del sistema para determinar las causas y los causantes de hechos no permitidos.

13.1.3. Tipos de seguridad en sistemas de computación:

Para proteger los sistemas de computación existen diferentes métodos dependiendo de lo que se quiere proteger. Así la seguridad de un sistema de computación puede dividirse en 3 tipos principales:

1. **Seguridad del Sistema:** Esta se enfoca en las características del sistema operativo que controlan quién puede accesar el sistema y los datos almacenados en éste. Incluye controles como contraseñas, auditorías del uso del sistema, respaldo y planes y políticas de seguridad.

2. **Seguridad de las Comunicaciones:** Se relaciona con la protección de la información mientras se transmite por cualquier medio fuera del sistema. Se enfoca en el acceso en red a los sistemas de computación y las tecnologías que aumentan la seguridad de los sistemas conectados a redes externas. Incluye controles como autenticación, encriptamiento y los *Rompefuegos (Firewalls)*.

3. **Seguridad Física:** Es la protección de los componentes físicos del sistema contra daños que pudieran ocasionarle agentes naturales o ataques premeditados.

13.1.4. Reglas de Sentido Común de Seguridad

La seguridad efectiva de un sistema tiene sus raices en el sentido común. A continuación se mencionan una serie de reglas que deben aplicarse. Nemeth, en su libro [4] compara el sentido común con el tratamiento que se le debe dar a los ratones en una casa.

- No deje cosas que pueden resultar interesantes para los ratones en la cocina. Queso, pan, etc. son muy atractivos para los ratones.

- Selle los huecos que los ratones utilizan para entrar en la casa.

- No proporcione lugares donde los ratones puedan construir sus nidos. Ropa sucia amontonada puede ser un buen nido.

- Coloque trampas para ratones en los sitios donde Ud los ha visto.

- Supervise las trampas diariamente, recárguelas y deságase de los ratones atrapados. Trampas llenas no capturan ratones.

- Evite el uso de venenos para lidiar con los ratones. Estos podrian dejar ratones muertos entre las paredes o matar a sus mascotas. Las trampas tradicionales son las mejores.

- Consiga un gato.

Ud. puede utilizar estas mismas reglas (ligeramente modificadas) para asegurar su sistema Unix.

- No coloque archivos que pueden ser interesantes a los hackers. Secretos de la compañia, nóminas, etc. deben ser manejados cuidadosamente. revise los comandos compress y crypt.

- Selle los huecos que pueden ser utilizados por los hackers para tener acceso al sistema. Revisar los boletines de seguridad con frecuencia puede ayudar. **(http://www.cert.org)**.

- No proporcione sitios donde los hackers puedan construir nidos en el sistema. Frecuentemente, los hackers entran en un sistema y lo utilizan como cuartel general para atacar a otros sistemas. Servidores ftp con directorios con permiso de escritura, cuentas con contraseñas débiles, cuentas de grupos, etc. son buenos sitios para contruir nidos.

- Coloque trampas básicas en los sistemas conectados a internet. Herramientas como tripwire, crack, tcpwrappers, etc. lo protegeran contra la plaga.

- Supervise los reportes generados por las herramientas. Un problema menor ignorado en un reporte puede llevar a un desastre más grande.

- Aprenda sobre seguridad en Unix. Un gran número de consultores de seguridad estarian felices de infringirle terror y decirle que por sólo 50 mil dólares pueden asegurar su sistema Unix. Usualmente estas soluciones le dejan un ratón muerto en su pared y merman la productividad de sus usuarios.

- Merodee en los alrededores en busca de actividades inusuales. Investigue sobre eventos sospechosos como cambios en la contabilidad, archivos log, etc.

13.1.5. Herramientas

Existe una gran variedad de herramientas que pueden ayudar a los administradores a fortalecer la seguridad en las redes y los elementos que las conforman. A continuación se mencionan algunas de ellas de dominio público.

- Crack es un programa que adivina contraseñas.

- Nessus descubre vulnerabilidades en diferentes sistemas.
- Snifit husmea el trafico de la red.
- Snort ayuda a detectar las huellas que dejan los ataques en los archivos del sistema.
- TCP_Wrapper es una herramienta que controla el acceso a los servicios de un sistema.
- Secure shell permite conexiones seguras entre sistemas remotos.
-

Selección de Password

Se estima que más del 80 % de los accesos no autorizados a sistemas ocurre debido al empleo de malos passwords, bien sea porque son descubiertos por fuerza bruta, porque son adivinados por tanteo o porque alguien "fisgonea" en el momento en que otro usuario introduce su password. Por esto es vital que se preste especial atención a la selección de password por parte de los usuarios.

Qué debe y qué no debe hacer un usuario al momento de elegir un password:

- NO utilice su login, nombre real, nombre del conyuge, de los hijos o de personas cercanas como password. Tampoco use su número de teléfono, de cédula o placa del carro. Tampoco les añada un número adelante o atrás a alguno de éstos.
- NO utilice secuencias de teclas como "qwerty.º "12345".
- NO utilice ninguna palabra escrita cerca de su lugar de trabajo.
- NO utilice ninguna palabra que aparezca en un diccionario en español, inglés u otro idioma susceptible de ser usado por Ud.
- NO use un password de menos de 5 caracteres ni de más de 8 caracteres.
- NO permita que nadie mire por encima de su hombro mientras Ud. introduce su password.
- NO escriba su password en ninguna parte, y mucho menos en una hoja de papel junto a su terminal o estación de trabajo; memorícelo.
- NO utilice ningún ejemplo de "buen"password que pueda ver en material al respecto.
- NO le diga a nadie su password, ni la regla que utiliza para construirlo (en caso de usar una forma especial de fabricarlo).

- USE combinaciones de letras minúsculas y mayúsculas, números y caracteres especiales (comas, espacios, comillas, paréntesis), pero cuide de que dichos caracteres especiales no sean tan .^es"peciales"que no aparezcan en todos los teclados, no sea que un día se encuentre frente a un teclado distinto del usual y no pueda entrar en su cuenta.
- CAMBIE el password tan frecuentemente como le sea posible.
- USE frases usuales, dichos, poemas o canciones para construir su password.

Un ejemplo de como construir un password: Tome una frase o estrofa facilmente memorizable y tome la inicial (o la segunda letra, o la última) de cada palabra para construir el password. Si tenemos:

Erase una vez hace mucho tiempo
password: e1v-Mt

Sustituye "una" por "1", la "hache" por "-" (porque es muda) y usa "M" mayúscula porque es hace MUCHO.

13.2. Supervisión del Sistema

El sistema de contabilidad, el kernel y varias utilidades producen información que es registrada periódicamente en el sistema. Eventualmente, esta información consume los recursos de almacenameinto principal, así que debe ser resumida, comprimida, archivada, y finalmente eliminada.

Diferentes políticas son utilizadas para la gestión de los archivos logs:

- Eliminar toda la información inmediatamente.
- Reiniciar la información periódicamente.
- Rotar los archivos logs, manteniendo la información por un tiempo fijo.
- Comprimir los archivos y almacenarlos en medios secundarios.

La selección de una política determinada depende de cuanto espacio de disco haya disponible. Sin embargo, cuando existe suficiente espacio en disco, lidiar con el crecimiento de estos archivos es tedioso. Una de las soluciones es utilizar el sistema automático de ejecución periódica de Unix para correr scripts de mantenimiento.

13.2.1. El archivo /etc/syslog.conf

En UNIX, los diferentes procesos pueden enviar mensajes de varios tipos y prioridades. Estos mensajes son captados por el proceso syslogd quién, de acuerdo a su configuración, puede almacenarlos localmente en archivos o enviarlos a otra máquina para que sean almacenados allí. Este archivo de configuración consiste en líneas de dos campos cada una. El primer campo es el campo de selección, donde se especifica el tipo de mensaje y la prioridad. El segundo campo es el campo de acción, donde se indica qué hacer al recibir un mensaje como el especificado en el campo primero. Ambos campos están separados por uno o más espacios o tabuladores.

El campo de selección se divide, a su vez, en dos partes separadas por un punto. Primero se especifica la facilidad y luego la prioridad. Existen las siguientes facilidades :

- auth : Mensajes relacionados con autorización y seguridad.
- authpriv: Mensajes relacionados con autorización y seguridad.
- cron : Mensajes del demonio del reloj (cron, at).
- daemon :Mensajes de otros demonios.
- kern : Reportes del kernel.
- lpr : Mensajes del sistema de impresoras.

- mail : Mensajes relacionados con el sistema de correos.
- mark : Mensaje generado periodicamente por el reloj.
- news : Mensajes relacionados con el sistema de USENET News.
- syslog : Mensajes internos del demonio syslogd.
- user :Mensajes genéricos.
- uucp : Mensajes relacionados con uucp.

Se tiene también varios niveles, de acuerdo a la severidad del mensaje :

- emerg : El sistema ha dejado de ser estable.
- alert : Problema, se requiere acción inmediata.
- crit : Condición crítica, pero puede seguir funcionando.
- err :Aviso de error.
- warning : Advertencia.
- notice : Condición normal pero importante.
- info : Mensaje de información.
- debug : Mensaje de depuración.

Si el nivel no es precedido de un signo de igual, se entiende que se trata de mensajes de ese nivel y superiores. El campo de acción puede ser de una de las cuatro formas siguientes :

- Un camino absoluto (precedido por / "): En ese caso los mensajes son colocados en el archivo indicado.
- Un nombre de máquina (precedido por "@ ") : El mensaje será enviado al syslogd de la máquina mencionada.
- Lista de usuarios separados por coma (",") : Los mensajes serán mostrados a esos usuarios si están conectados.
- Un asterisco : Los mensajes se le mostrarán a todos los usuarios conectados.

13.2.2. El Comando last

Uno de los comandos más utilizados para supervisar la entrada de usuarios es el comando **last**. Este despliega la lista de usuarios que han entrado y salido del sistema desde que el archivo log que maneja esta información fue creado. Ejemplo:

```
#last
jdiaz    tty3                          Fri May 11 11:34   still logged in
jdiaz    tty2                          Fri May 11 11:34   still logged in
jdiaz    tty1                          Fri May 11 11:17   still logged in
reboot   system boot  2.2.16-22        Fri May 11 11:16          (00:18)
root     tty2                          Fri May 11 10:17 - down   (00:00)
jdiaz    tty1                          Fri May 11 08:26 - 09:27  (01:00)
gilberto pts/1        hydra0.cecalc.ul Fri May 11 08:26 - 10:17  (01:51)
gilberto pts/0        hydra0.cecalc.ul Fri May 11 08:26 - 10:17  (01:51)
root     tty1                          Fri May 11 08:25 - 08:25  (00:00)
reboot   system boot  2.2.16-22        Fri May 11 08:18          (01:58)
jdiaz    tty3                          Thu May 10 19:54 - 20:54  (01:00)
jdiaz    tty2                          Thu May 10 19:54 - 20:55  (01:01)
jdiaz    tty1                          Thu May 10 19:37 - down   (04:00)
reboot   system boot  2.2.16-22        Thu May 10 19:36          (04:01)
```

13.2.3. El archivo messages

Este archivo, generalmente ubicado en el directorio /var o en alguno de sus subdirectorios, contiene información sobre muchos de los eventos ocurridos en el sistema, por ejemplo, conexiones ftp, reinicios del sistema, entradas al sistema, etc.

```
Nov  6 05:08:37 odie kernel: sb: No ISAPnP cards found, trying standard ones...
Nov  6 05:08:37 odie kernel: SB 3.02 detected OK (220)
Nov  6 05:08:37 odie kernel: This sound card may not be fully Sound Blaster Pro
compatible.
Nov  6 05:08:37 odie kernel: In many cases there is another way to configure OSS so that
Nov  6 05:08:37 odie kernel: it works properly with OSS (for example in 16 bit mode).
Nov  6 05:08:37 odie kernel: Please ignore this message if you _really_ have a SB Pro.
Nov  6 05:10:32 odie ftpd[2041]: FTP LOGIN FROM 150.185.138.100 [150.185.138.100], jdiaz
Nov  6 05:11:07 odie ftpd[2041]: FTP session closed
Nov  6 05:11:54 odie kernel: eth0: lost link beat
Nov  6 05:11:54 odie kernel: eth0: autonegotiation restarted
Nov  6 05:13:59 odie apmd[567]: Now using AC Power
Nov  6 05:14:23 odie login(pam_unix)[2033]: session opened for user jdiaz by LOGIN(uid=0)
Nov  6 05:14:23 odie  -- jdiaz[2033]: LOGIN ON tty1 BY jdiaz
```

13.2.4. Herramientas de Supervisión de Red

El comando netstat -a

Este comando, discutido en secciones anteriores, tiene un opción útil (-a) para la supervisión de las conexiones actuales del sistema con clientes o servidores remotos. Ejecutelo y observe su salida.

El comando lsof

El comando **lsof** al ser ejecutado despliega todos los archivos del sistema que estan abiertos y el proceso correspondiente que los utiliza

```
COMMAND   PID     USER   FD   TYPE DEVICE    SIZE     NODE NAME
bash     1926 gilberto  cwd    DIR    3,5    2192    79261 /home1/gilberto/
bash     1926 gilberto  rtd    DIR    3,3     664        2 /
bash     1926 gilberto  txt    REG    3,3  671628  1960237 /bin/bash
bash     1926 gilberto  mem    REG    3,3 1237276  6958828 /lib/libc-2.6.1.so
bash     1926 gilberto  mem    REG    3,3    9612  6958980 /lib/libdl-2.6.1.so
bash     1926 gilberto  mem    REG    3,3  108996  6959000 /lib/ld-2.6.1.so
bash     1926 gilberto   0u    CHR  136,0                2 /dev/pts/0
bash     1926 gilberto   1u    CHR  136,0                2 /dev/pts/0
bash     1926 gilberto   2u    CHR  136,0                2 /dev/pts/0
```

Bibliografía

[1] Marco Cesati Daniel P. Bovet. *Understanding the Linux Kernel.* O'Reilly & Associates Inc., 2003.

[2] Quarterman J. *The Matrix: Computer Networks and Conferencing Systems Worldwide.* Digital Press, United States of America, 1990.

[3] Gene Spafford Simson Garfinkel, Alan Schwartz. *Practical Unix & Internet Security, 3rd.* O'Reilly & Associates, Inc., United States of America, February 2003.

[4] Nemeth Evi; Snyder Garth; Seebass Scott; Hein Trent. *Unix System Administration Handbook.* Prentice Hall, 2000.