

TUTORIAL 1

INTRODUCCIÓN AL USO DE LINUX

Prof. Ing. Miguel Angel Aguilar Ulloa

© 2009-2010





Usted es libre de:

✓ Copiar, distribuir y comunicar públicamente la obra.



✓ Hacer obras derivadas.

Bajo las siguientes condiciones:



✓ Reconocimiento — Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciador (pero no de una manera que sugiera que tiene su apoyo o apoyan el uso que hace de su obra).



✓ No comercial — No puede utilizar esta obra para fines comerciales.



✓ Compartir bajo la misma licencia — Si altera o transforma esta obra, o genera una obra derivada, sólo puede distribuir la obra generada bajo una licencia idéntica a ésta.

Texto de la licencia: <http://creativecommons.org/licenses/by-nc-sa/3.0/legalcode>.



1. ¿Qué es Linux?
2. Historia.
3. Distribuciones.
4. Líneas de comandos (shells).
5. Ayuda sobre los comandos.
 1. Obtener ayuda sobre los comandos.
 2. Acceso a las páginas de manuales e información.
6. Sistema de archivos y manejo de archivos.
 1. Estructura de un sistema de archivos Linux.
 2. Todo es un archivo.
 3. Manejo de archivos y directorios.
 4. Despliegue, escaneo y ordenamiento de archivos.
 5. Enlaces simbólicos y duros.
 6. Derechos de acceso a los archivos.

7. Entrada/Salida estándar, redirecciones y pipes.
 1. Entrada y salida estándar, redireccionamiento a archivos.
 2. Error estándar.
 3. Pipes: Redirigiendo la salida estándar a otros comandos.
8. Control de tareas.
 1. Control total de tareas.
 2. Ejecución en segundo plano, suspender y resumir.
 3. Lista de tareas activas.
 4. Matar procesos.
 5. Variables de ambiente.
 6. Variables de ambiente PATH.
 7. Alias de shell y archivo .bashrc.

9. Misceláneas

1. Editores de texto.
2. Compresión y archivamiento.
3. Impresión de archivos.
4. Comparación de archivos y directorios.
5. Buscar archivos.

10. Bases de la administración del sistema

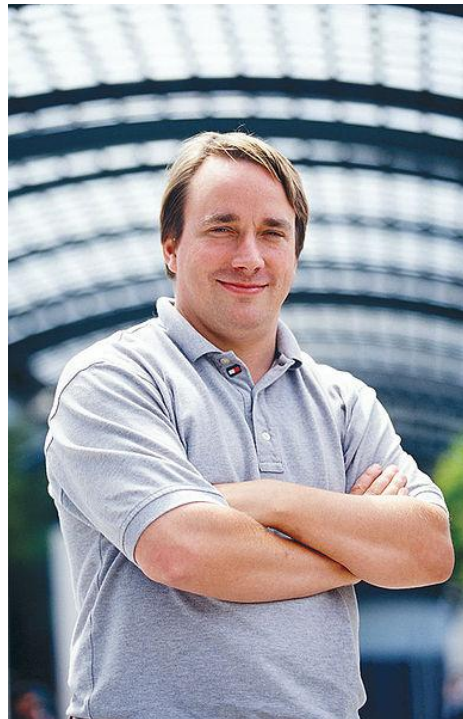
1. Uso de usuario root.
2. Manejo de usuarios.
3. Configuración de la red.
4. Crear y montar sistemas de archivos.
5. Manejo de paquetes.

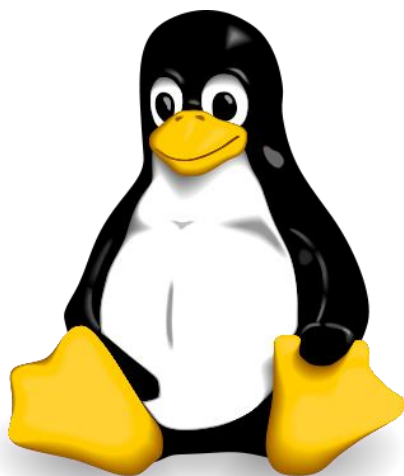
11. SSH

1. Instalación y uso básico.
2. Transferencia de archivos y redireccionamiento de X.
3. Ejecución remota.
4. Evitar el password con el uso de llaves.

1. ¿QUÉ ES LINUX?

- *Linux* es el núcleo o *kernel* del sistema operativo libre GNU/Linux. Lanzado bajo la licencia pública general de GNU y desarrollado gracias a contribuciones provenientes de todo el mundo, Linux es uno de los ejemplos más notables de software libre.
- Linux fue creado por Linus Torvalds en 1991. Muy pronto, la comunidad de Minix (un clon del sistema operativo Unix) contribuyó en el código y en ideas para el núcleo Linux.





2. HISTORIA DE LINUX

- En 1991, en Helsinki, Linus Torvalds comenzó un proyecto que más tarde se llegó a ser el núcleo Linux. Esto fue al principio un emulador terminal, al cual Torvalds solía tener acceso en los grandes servidores UNIX de la universidad.
- Él escribió el programa expresamente para el hardware que usaba, e independiente de un sistema operativo, porque quiso usar las funciones de su nueva computadora personal con un procesador 80386. Este es aún el estándar actual. El sistema operativo que él usó durante el desarrollo fue Minix, y el compilador inicial fue el GNU C, que aún es la opción principal para compilar Linux (aunque Linux puede ser compilado bajo otros compiladores, tal como el Intel C Compiler).

El 25 de agosto de 1991, él anunció su creación inicial en el grupo de noticias MINIX *comp.os.minix*:

Message-ID: 1991Aug25.205708.9541@klaava.helsinki.fi
From: torvalds@klaava.helsinki.fi (Linus Benedict Torvalds)
To: Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system

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. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-sytem due to practical reasons)among other things.

I've currently ported bash (1.08) an gcc (1.40), and things seem to work. This implies that i'll get something practical within a few months, and I'd like to know what features most people want.

Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus Torvalds torvalds@kruuna.helsinki.fi

3. DISTRIBUCIONES DE LINUX

- Una distribución Linux o distribución GNU/Linux (coloquialmente llamadas *distros*) es cada una de las variantes de este sistema operativo que incorpora determinados paquetes de software para satisfacer las necesidades de un grupo específico de usuarios, dando así origen a ediciones domésticas, empresariales y para servidores.
- Las distribuciones son ensambladas por individuos, empresas u otros organismos.
- Cada distribución puede incluir cualquier número de software adicional, incluyendo software que facilite la instalación del sistema. La base del software incluido con cada distribución incluye el núcleo Linux y las herramientas GNU, al que suelen añadirse también varios paquetes de software.
- Las herramientas que suelen incluirse en la distribución de este sistema operativo se obtienen de diversas fuentes, y en especial de proyectos de código abierto o software libre.

Debian: una distribución mantenida por una red de desarrolladores voluntarios con una gran compromiso por los principios del software libre.

Fedora: una distribución lanzada por Red Hat para la comunidad.

Gentoo: una distribución orientada a usuarios avanzados, conocida por la similitud en su sistema de paquetes con el FreeBSD Ports, un sistema que automatiza la compilación de aplicaciones desde su código fuente.

Knoppix: la primera distribución live en correr completamente desde un medio extraíble. Esta basada en Debian.

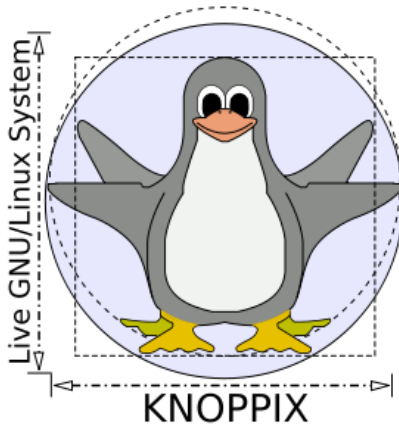
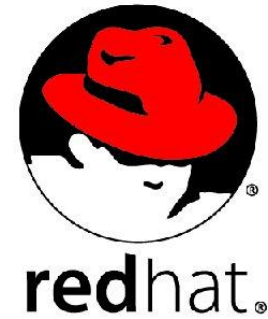
Kubuntu: la versión en KDE de Ubuntu.

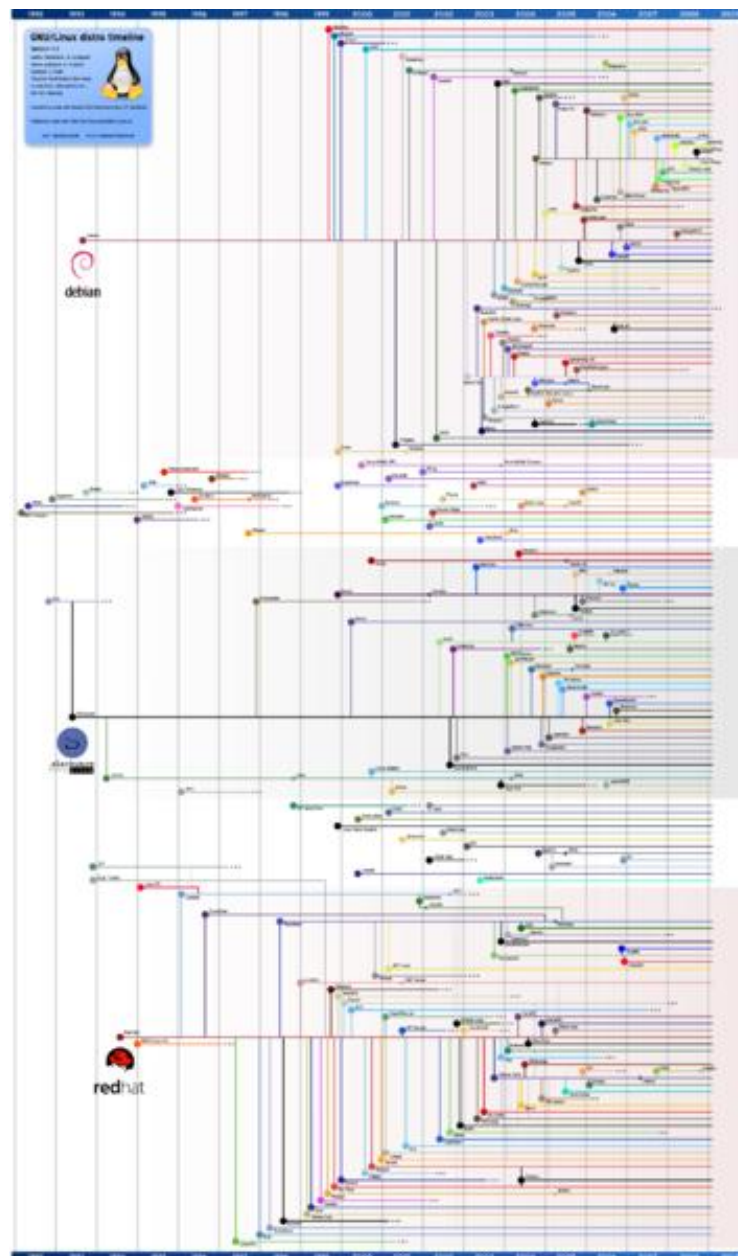
Mandriva: mantenida por la compañía francesa del mismo nombre, es un sistema popular en Francia y Brasil. Esta basada en Red Hat.

openSUSE: originalmente basada en Slackware es patrocinada actualmente por la compañía Novell.

Red Hat Enterprise Linux: derivada de Fedora, es mantenida y soportada comercialmente por Red Hat.

Ubuntu: una popular distribución para escritorio basada en Debian y es considerada la más sencilla para usuarios principiantes.





4. LÍNEAS DE COMANDOS (SHELLS)

- Shells: herramientas para ejecutar comandos de usuario.
- Se le llama “shells” debido a que esconden los detalles del sistema.
- Los comandos son entradas en una terminal de texto, tanto en una ventana en un ambiente gráfico, como en una consola.
- Los resultados son desplegados en la terminal.
- Las shells pueden interpretar scripts, proveen todos los recursos para escribir programas complejos (condicionales, ciclos, variables, etc).

✓ *sh - Bourne Shell (obsoleta):*

Tradicional, shell básico encontrado en sistemas UNIX, creada por Steve Bourne.

✓ *csh – La shell de C (obsoleta):*

Fue un shell popular con una sintaxis similar a C.

✓ *tcsh – La shell TC (sigue siendo popular):*

Es una implementación de un shell compatible con C y con características como: completado de comandos, historial, etc.

✓ *bash – El shell Bourne de nuevo (más popular):*

Es una versión mejorada de sh con muchas características adicionales.

5. AYUDA SOBRE LOS COMANDOS

- Algunos comandos UNIX y la mayoría de los comandos GNU / Linux ofrecen al menos un argumento de ayuda.

✓ **-h**

- es mayormente usada para introducir opciones de un caracter.

✓ **--help**

- es siempre empleada para introducir el nombre completo de la opción, lo que hace los scripts más fáciles de leer.

También es posible obtener un pequeño resumen de las opciones cuando se ingresa un argumento inválido.

✓ `man <palabra clave>`

Despliega una o varias páginas de manual para `<palabra clave>`.

✓ `man man`

La mayoría de páginas de manual son acerca de comandos de UNIX, pero las hay también acerca de funciones de C, de encabezados de bibliotecas (*.h) i estructuras de datos o inclusive archivos de configuración del sistema.

✓ `man stdio.h`

✓ `man fstab` (para `/etc/fstab`)

Las páginas de manuales se ubican en los directorios especificados en la variable de ambiente llamada `MANPATH`.

En GNU, las páginas de manuales están siendo reemplazadas por las páginas de información. Algunas páginas de manuales inclusive hacen referencia a las páginas de información.

✓ **info** <comando>

Características de **info**:

- ✓ Documentación estructurada en secciones y subsecciones.
- ✓ Posibilidad de navegar en la estructura.
- ✓ Las páginas de información son generadas con el mismo código fuente de las páginas de documentación en HTML.

6. SISTEMA DE ARCHIVOS Y MANEJO DE ARCHIVOS

✓ Los archivos regulares.

✓ Directorios.

Los directorios son solo archivos que listan un conjunto de archivos.

✓ Enlaces simbólicos.

Archivos que funciona como accesos directos a otros archivos.

✓ Dispositivos y periféricos.

Se lee y se escribe de los dispositivos como archivos regulares.

✓ Pipes

Usados para hacer cascadas de programas.

```
cat *.log | grep error
```

✓ Sockets

Comunicación inter-proceso (IPC).

- ✓ Son sensibles a mayúsculas.
- ✓ Pueden tener cualquier carácter, incluyendo el espacio en blanco y exceptuando / .

Los tipos de archivos en los propios archivos como metainformación.

Las extensiones en el nombre de los archivos no son necesarias ni interpretadas. Solo son empleadas por conveniencia.

Sistema de archivos unix

- ✓ Ejemplos de nombres de archivos:

README

.bashrc

Buglist

Index.html

Makefile

host.required

Una ruta es una secuencia de directorios anidados con un archivo o directorio al final, y separados por el caracter/.

✓ Ruta relativa: `archivos/libros/linux.pdf`

Relativo al directorio actual.

✓ Ruta absoluta:

`/home/maguilar/archivos/libros/linux.pdf`

✓ Directorio raíz: `/`

Es el inicio de todas las rutas absolutas en el sistema.

No es una estructura definitiva, puede cambiar entre un sistema y otro.

<code>/</code>	Directorio de raíz.
<code>/bin/</code>	Comandos esenciales del sistema.
<code>/boot/</code>	Imágenes del kernel.
<code>/dev/</code>	Archivos que representan dispositivos. Ej. <code>/dev/ttyS0</code> : puerto serial.
<code>/etc/</code>	Archivos de configuración del sistema.
<code>/home/</code>	Archivos de usuario.
<code>/lib/</code>	Bibliotecas compartidas básicas del sistema.
<code>/lost+found/</code>	Archivos corruptos que el sistema trató de recuperar.
<code>/media/</code>	Puntos de montaje para dispositivos removibles <code>/media/usbdisk</code> , <code>/media/cdrom</code>
<code>/mnt/</code>	Puntos de montaje para sistemas de archivos temporales.

<code>/opt/</code>	Herramientas específicas instaladas por el administrador del sistema. A veces se usa en lugar de <code>/usr/local</code> .
<code>/proc/</code>	Información del sistema. <code>/proc/cpuinfo</code> , <code>/proc/version</code>
<code>/root/</code>	Directorio del usuario root.
<code>/sbin/</code>	Comandos del administrador.
<code>/sys/</code>	Controles del sistema y de dispositivos. Frecuencia del CPU, consumo de poder de los dispositivos
<code>/tmp/</code>	Archivos temporales.
<code>/usr/</code>	Herramientas regulares del usuario, no esenciales para el sistema. <code>/usr/bin/</code> , <code>/usr/lib/</code> , <code>/usr/sbin/...</code>
<code>/usr/local/</code>	Software específico instalado por el administrador del sistema.
<code>/var/</code>	Datos usados por el sistema o servidores. <code>/var/log/</code> , <code>/var/spool/mail/</code> , <code>/var/www/</code> .

Lista todos los archivos en el directorio actual, en un orden alfanumérico, excepto por los archivos que comienzan con el carácter “.”.

✓ **ls -a** (all)

Lista todos los archivos, incluyendo los ocultos (*).

✓ **ls -S** (size)

Lista los archivos más grandes primero.

✓ **ls -l** (long)

Lista la información completa de los archivos: tipo, fecha, tamaño, propietario, permisos.

✓ **ls -r** (reverse)

Revuelve el ordenamiento de los archivos.

✓ **ls -t** (time)

Lista los archivos más recientes.

✓ **ls -ltr** (combinación de opciones)

Lista larga, y archivos más recientes al final.

✓ `ls *txt`

El shell primero reemplaza `*txt` por todos los nombres de archivos y directorios que terminen en `txt` (incluyendo `.txt`), y luego ejecuta el comando `ls`.

✓ `ls -d .*`

Lista todos los archivos y directorios que empiecen con `.`, `-d` le dice a `ls` que no despliegue el contenido de los directorios.

✓ `cat ?.log`

Despliega todos los archivos cuyos nombres empiecen por un carácter y terminen con `.log`.

✓ `./`

Es el directorio actual. Es útil para comando que toman el directorio como argumento. También es útil para comandos que se ejecutan en el directorio actual.

Así `./readme.txt` y `readme.txt` son equivalentes.

✓ `../`

Directorio padre. Uso típico `cd ..`.

✓ `~/`

No es un directorio especial de hecho. El shell solo lo substituye por el directorio home del usuario actual.

✓ `cd <directorio>`

Cambia el directorio actual a <directorio>.

✓ `cd -`

Regresa al directorio anterior.

✓ `pwd`

Despliega el directorio actual.

✓ **cp <archivo_fuente> <archivo_objetivo>**

Copia el archivo fuente al objetivo.

✓ **cp <archivo1> <archivo2>... <directorio>**

Copia los archivos al directorio objetivo (último argumento).

✓ **cp -i**

Pregunta al usuario por confirmación si el archivo objetivo existía previamente.

✓ **cp -r <directorio_fuente> <directorio_objetivo>**

Copia el directorio fuente al directorio objetivo. Se le conoce como copia recursiva.

rsync (remote sync) fue diseñado para mantener sincronizados dos directorios o máquinas.

- ✓ Solo copia los archivos que han cambiado. Los archivos del mismo tamaño son comparados mediante checksums.
- ✓ Solo transfiere los bloques que difieren dentro de los archivos.
- ✓ Puede comprimir los bloques que serán transferidos.
- ✓ Preserva los enlaces simbólicos y los permisos del archivo.
- ✓ Pueden trabajar a través de ssh (secure remote shell). Muy útil para actualizar los contenidos de una página web, por ejemplo.

✓ `rsync -a /home/maguilar/ /home/invitado/`

`-a` (archive mode). Es el modo más sencillo.

✓ `rsync -Pav --delete /home/maguilar/
/home/invitado/`

`-P: --partial` mantiene los archivos parcialmente transferidos.

`--progress` muestra el progreso durante la transferencia.

`--delete` borra los archivos en el directorio objetivo que no existen en el origen.

Cuidado: los nombres de los directorios deben terminar con `/`, o de otra manera obtendrá `/home/invitado/maguilar` dentro del directorio de destino.

```
✓ rsync -Pav /home/maguilar/documentos/ \
maguilar@dominio.com:/home/maguilar/documentos/
```

Copia a una máquina remota. Se preguntará el password para maguilar en el servidor.

```
✓rsync -Pav -e ssh \
maguilar@dominio.com:/home/maguilar/documentos/ \
/home/maguilar/documentos/
```

Copia de una máquina remota por ssh. Se preguntará por el password de maguilar.

✓ **mv <nombre_viejo> <nuevo_nombre>** (move)

Cambiar de nombre a un archivo o directorio.

✓ **mv -i**

Pregunta por confirmación al usuario si el nuevo archivo ya existía.

✓ **rm -i** (remove)

Pregunta por confirmación de usuario siempre que va a borrar un archivo o directorio.

✓ **rm -r <directorio1> <directorio2>** (recursive)

Remueve los directorios dados con sus contenidos.

✓ `mkdir <directorio1> <directorio2>` (make dir)

Crea directorios con los nombres dados.

✓ `rmdir <directorio1> <directorio2>` (remove dir)

Remueve los directorios dados. Solo funciona con directorios limpios. El comando alternativo es `rm -r`.

✓ **cat** <archivo1> <archivo2> ... (concatenate)

Concatena y muestra la salida del contenido de los archivos dados.

✓ **more** <archivo1> <archivo2> ...

Después de cada página le pide al usuario que presione una tecla para continuar. Puede brincar a palabras claves con el comando **/<palabra_clave>** .

✓ **less** <archivo1> <archivo2> ...

Hace más que more.

No lee el archivo entero antes de empezar.

Soporta el movimiento hacia atrás en el archivo con el mando **?** .

✓ `head [-<n>] <archivo>`

Muestra las primeras <n> líneas (o 10 por defecto) del archivo dado. No tiene que abrir el archivo completo para hacer esto.

✓ `tail [-<n>] <archivo>`

Muestra las últimas <n> líneas (o 10 por defecto) del archivo dado. No necesita cargar el archivo completo.

✓ `tail -f <archivo>` (follow)

Despliega las últimas 10 líneas del archivo dado y continua desplegando nuevas líneas que son agregadas al archivo dinámicamente.

Esto es muy útil cuando se está revisando archivos de registro que están cambiando continuamente.

✓ **grep <patrón> <archivos>**

Escanea los archivos dados y muestra las líneas que coinciden con el patrón dado.

✓ **grep error *.log**

Despliega todas las líneas que contienen error en los archivos ***.log**.

✓ **grep -i error *.log**

Lo mismo pero no sensible a mayúsculas.

✓ **grep -ri error .**

Lo mismo que el anterior pero recursivo en todos los archivos en el directorio actual (.) y sus subdirectorios.

✓ **grep -v info *.log**

Muestra todas líneas excepto las que contienen **info**.

✓ `sort <archivo>`

Ordena por caracteres las líneas en el archivo dado y muestra la salida.

✓ `sort -r <archivo>`

Lo mismo pero en orden inverso.

✓ `sort -ru <archivo>`

`-u`: unique. Lo mismo pero solo muestra la salida de líneas idénticas una vez.

Un enlace simbólico es un archivo especial el cual solo hace referencia a otro archivo en cuanto a nombre y directorio.

✓ Es muy útil para reducir el uso de disco cuando dos archivos o directorios tienen el mismo contenido.

✓ Como identificar los enlaces simbólicos:

1. `ls -l` muestra con `->` el archivo enlazado.
2. `ls` muestra el enlace con un color diferente.

✓ `ln -s <archivo> <nombre_enlace>`

Crea un enlace simbólico.

✓ `ln -s ../<archivo>`

Crea el enlace que está en otro directorio en el directorio actual con el mismo nombre.

✓ `ln -s <archivo1> <archivo2> ... <directorio>`

Crea múltiples enlaces al mismo tiempo en el directorio dado.

✓ `rm <nombre_enlace>`

Borra el enlace pero no el archivo enlazado propiamente.

- El comportamiento por defecto de ln es crear enlaces duros.
- Un enlace duro a un archivo es un archivo regular con el contenido exacto del archivo original.
- Los enlaces duros no pueden ser distinguidos de los archivos originales.
- Si se remueve el archivo original no hay impacto en el contenido del enlace duro.
- El contenido es removido cuando no hay más archivos (enlaces duros) hacia ellos.

Usuarios

Interface de nombre de archivos

Enlace simbólico

Archivo

Enlace duro

Interface de inodos

Inodo

- Use `ls -l` para verificar el derecho de acceso a los archivos.

Hay tres tipos de permisos de accesos.

1. Acceso de lectura (**r**)
2. Acceso de escritura (**w**)
3. Permiso de ejecución (**x**)

Hay tres tipos de niveles de acceso:

1. Usuario (**u**): para el propietario del archivo
2. Grupo (**g**): cada archivo tiene también un atributo de grupo, correspondiente a la lista dada de usuarios.
3. Otros (**o**): para todos los otros usuarios.

- **x** sin **r** es legal pero inutil.

Se tiene que tener permisos de lectura para poder ejecutar un archivo.

- Tanto **r** como **x** son necesario para los directorios.

x para entrar y **r** para mostrar su contenido.

- No se puede renombrar, remover, copiar archivos en un directorio si no se tiene el permiso **w** para éste directorio.

-

Ejemplos de permisos de acceso

✓ **-rw-r--r--**

El propietario del archivo lo puede leer y escribir, y el resto solo lo pueden leer.

✓ **-rw-r-----**

El propietario del archivo lo puede leer y escribir, y los miembros del grupo lo pueden leer.

✓ **drwx-----**

Directorio que es solo accesible por su propietario.

✓ **-----r-x**

Archivo que solo usuarios que no pertenezcan al grupo ni siquiera su propietario lo pueden leer.

✓ `chmod <permisos> <archivos>`

Hay dos formatos para los permisos.

1. Formato octal (abc)

`a, b, c = r * 4 + w * 2 + x` (`r, w, x`: booleanos).

Ejemplo: `chmod 644 <archivo>`

(`rw` para `u`, `r` para `g` y `o`).

2. Formato simbólico: fácil de entender y más explícito.

`chmod go+r`: agrega permisos de lectura al grupo y al resto.

`chmod u-w`: quita los permisos de escritura al propietario.

`chmod a-x`: (`a`:all), quita los permisos de ejecución para todos.

`chmod -R a+r`: agrega permisos de lectura para todos recursivamente.

Particularmente útil en sistemas empotrados cuando se crean archivos para otro sistema.

✓ `chown -R maguilar /home/linux/src` (-R: recursive)

Hace que **maguilar** sea el nuevo propietario de todos los archivos en el directorio dado.

✓ `chgrp -R curso /home/linux/src`

Hace a curso el nuevo grupo de los archivos en el directorio dado.

✓ `chown -R maguilar:curso /home/linux/src`

Permite cambiar el grupo y el propietario al mismo tiempo.

7. ENTRADA/SALIDA ESTÁNDAR, REDIRECCIONES Y PIPES

- Todas las salidas de texto de los comandos en la terminal se realizan a través de su salida estándar.
- La salida estándar puede ser redirigida a un archivo empleando el símbolo **>**.
- La salida estándar puede ser agregada a un archivo preexistente usando el símbolo **>>**.
- Ejemplos:
 - ✓ `ls /home/maguilar > home.txt`
 - ✓ `cat home.txt > contenido_directorios.txt \`
`cat tmp.txt >> contenido_directorios.txt`
 - ✓ `echo "hola" > hola.txt`

- Muchos comandos pueden tomar sus entradas de la entrada estándar cuando no reciben entradas de sus argumentos.

- Ejemplos:

✓ `sort`

`windows`

`linux`

`[ctrl][D]`

`linux`

`windows`

✓ `sort < participantes.txt`

La entrada estándar es tomada del archivo dado.

- Los pipes de UNIX son muy útiles para redirigir la salida estándar de un comando a la entrada estándar de otro.

- Ejemplos:

- ✓ `cat *.log | grep error | sort`

- ✓ `grep -ri error . | grep -v "ignorado" | sort -u \`
`> errores_serios.log`

- ✓ `cat /home/*/homework.txt | grep ejercicio1 | more`

✓ `tee [-a] <archivo>`

El comando `tee` puede ser usado para enviar la salida estándar a la pantalla y un archivo simultáneamente.

✓ `make | tee build.log`

Ejecuta el comando `make` y almacena la salida en `build.log`.

✓ `make install | tee -a build.log`

Ejecuta el comando `make install` y agrega su salida a `build.log` sin borrar el contenido previo de `build.log`.

- Son mensajes de error, usualmente salidas (si el programa esta bien escrito) que se envían al error estándar en lugar de a la salida estándar.
- El error estándar puede ser redirigida mediante **2>** o **2>>**.
- Ejemplo:

```
✓ cat a1 a2 archivo_no_existente > a_nuevo \  
2> a_error
```

- Nota: 1 es el descriptor para la salida estándar, así que 1> es equivalente a >.
- Para redirigir tanto la salida estándar como el error estándar al mismo archivo se usa &> :

```
✓ cat a1 a2 archivo_no_existente &> archivo_salida
```

- Es útil para llenar la entrada estándar siempre con el mismo texto.

✓ `yes <texto> | <comando>`

Se mantiene llenando la entrada estándar de <comando> con <texto> (y por defecto).

- Ejemplos:

✓ `yes | rm -r dir/`

✓ `yes "" | make oldconfig`

Es equivalente a presionar enter para aceptar la configuración por defecto.

- Dispositivos con un comportamiento o contenido especial.

✓ `/dev/null`

Es un vertedero de datos. Se usa para mandar mensajes no deseados de la salida estandar o del error estandar.

```
make &> /dev/null
```

✓ `/dev/zero`

Las lecturas a éste archivo retornan siempre caracteres `\0`.

Es útil para crear archivos llenos de ceros.

Revise `man null` o `man zero` para detalles.

✓ `/dev/random`

Retorna bytes aleatorios cuando es leído. Usa interrupciones de algún dispositivo para crear una verdadera aleatoriedad.

✓ `/dev/urandom`

Genera bytes pseudo aleatorios.

Revise `man random` para detalles.

8. CONTROL DE TAREAS

“Todo en UNIX es un archivo. Todo en UNIX que no es un archivo es un proceso.”

- Procesos:

- ✓ Instancias de programas en RAM.

- ✓ Varias instancias del mismo programa pueden ser ejecutadas al mismo tiempo.

- ✓ Datos asociados a procesos:

Archivos abiertos, memoria reservada, pila, id del proceso, prioridad, estado, etc.

- Desde el principio, UNIX soporta procesamiento multitarea preemptivo real.
- Habilidad de ejecutar muchas tareas en paralelo, y abortarlas inclusive si ellas corrompen su estado y sus datos.
- Habilidad de escoger que programas son ejecutados.
- Habilidad de escoger que entrada se dirige a los programas y hacia donde va su salida.

Es útil para:

- ✓ Para tareas de línea de comandos cuya salida puede ser examinada después, especialmente para las que consumen mucho tiempo.
- ✓ Para empezar aplicaciones gráficas desde la línea de comando y luego continuar con el mouse.

Para empezar un tarea en segundo plano se agrega **&** al final de comando.

✓ **`./instalador &`**

✓ `jobs`

Retorna la lista de tareas en segundo plano de la misma consola.

```
[1]- Running ~/bin/instalador &
```

```
[2]+ Running make &
```

✓ `fg %<n>`

Pone la última o la nésima tarea de segundo plano en primer plano.

✓ `[Ctrl] Z` `bg`

Mueve la tarea actual a un segundo plano.

✓ `kill %<n>`

Mata la nésima tarea.

```
> jobs
```

```
[1]-  Running ~/bin/instalador &
```

```
[2]+  Running make &
```

```
> fg
```

```
make
```

```
> [Ctrl] Z
```

```
[2]+  Stopped make
```

```
> bg
```

```
[2]+  make &
```

```
> kill %1
```

```
[1]+  Terminated ~/bin/instalador
```

Lista de todos los procesos

✓ **ps -ux**

Lista todos los procesos que pertenecen al usuario actual.

✓ **ps -aux**

Lista todos los procesos del sistema.

✓ **ps aux | grep maguilar | grep bash**

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
maguilar	3039	0.0	0.2	5916	1380	pts/2	S	14:35	0:00	/bin/bash
maguilar	3190	0.0	0.2	6368	1360	pts/4	R	14:37	0:00	/bin/bash

PID: Id del proceso.

VSZ: Tamaño de la memoria virtual del proceso.

RSS: Tamaño del proceso en la RAM.

TTY: Terminal.

STAT: Estatus: R(Ejecutándose), S (Dormido), W(paginando), Z (zombie).

✓ top

Muestra los procesos más importantes ordenados por el consumo de cpu.

top 15:

44:33 up 1:11, 5 users, load average: 0.98, 0.61, 0.59

Tasks: 81 total, 5 running, 76 sleeping, 0 stopped, 0 zombie

Cpu(s): 92.7% us, 5.3% sy, 0.0% ni, 0.0% id, 1.7% wa, 0.3% hi, 0.0% si

Mem: 515344k total, 512384k used, 2960k free, 20464k buffers

Swap: 1044184k total, 0k used, 1044184k free, 277660k cached

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3809	jdope	25	0	6256	3932	1312	R	93.8	0.8	0:21.49	bunzip2
2769	root	16	0	157m	80m	90m	R	2.7	16.0	5:21.01	X
3006	jdope	15	0	30928	15m	27m	S	0.3	3.0	0:22.40	kdeinit
3810	jdope	16	0	2892	916	1620	R	0.3	0.2	0:00.06	top

✓ Es posible cambiar el ordenamiento: **M**: uso de memoria, **P**: % de CPU, **T**: Tiempo.

✓ Es posible matar una tarea escribiendo k y el id del proceso.

✓ `kill <idsp>`

Envía un señal de abortar a los procesos dados. Le permite a los procesos salvar los datos y salir por ellos mismos.

```
kill 3039 3134 3190
```

✓ `kill -9 <idsp>`

Envía una señal de terminación inmediata. El sistema en sí mismo termina los procesos. Es útil cuando los procesos están realmente pegados.

✓ `kill -9 -1`

Mata todos los procesos del usuario actual. `-1`: significa todos los procesos.

✓ `killall [-<señal>] <comando>`

Mata todos los procesos `<comando>`. Ejemplo: `killall instalador`

✓ `xkill`

Le permite matar una aplicación gráficamente con solo darle click.

- ✓ Cuando una interfaz gráfica se pega, no reinicie el sistema.
- ✓ Es muy común que el sistema esté aún vivo. Intente acceder a una consola de texto con `[CTRL] [Alt] [F1]`, o `[F2] [F3]` para más consolas.
- ✓ En la consola de texto se puede intentar matar la aplicación problemática.
- ✓ Una vez que mato el proceso, se puede ir de regreso a la sesión gráfica con `[CTRL] [ALT] [F5]` o `[F7]`, dependiendo de la distribución.
- ✓ Si no puede identificar el proceso problemático puede matarlo con `kill -9 -1`.

- Es posible ejecutar secuencias de comandos en una sola línea de la consola. Para ellos se utiliza el símbolo `;`.

```
echo "Hola"; sleep 10; echo " ... Adios"
```

- Condicionales: use `||` (or) o `&&` (and):

```
✓ more instalar || echo "La instalación falló"
```

Se ejecuta `echo` solo si el primer comando falla.

```
✓ ls recetas && cat recetas/* > recetas.txt
```

Solo ejecuta `cat` si el comando `ls` es exitoso.

Comillas dobles (") pueden ser utilizadas para prevenir que el shell interprete los espacios como separadores de argumentos, así como para prevenir la expansión de los patrones de nombres de archivos.

```
✓echo "Hola Mundo"
```

```
Hola Mundo
```

```
✓echo "Usted está registrado como $USER"
```

```
Usted está registrado como maguilar
```

```
✓echo *.log
```

```
Instalacion.log errores.log
```

```
✓echo "*.log"
```

```
*.log
```

Las comillas simples brindan una funcionalidad similar, pero lo que están entre las comillas no es sustituido.

```
✓ echo "Usted está registrado como $USER"  
Usted está registrado como $USER
```

Las comillas inversas (`) pueden ser empleadas para llamar un comando dentro de otro.

```
✓ cd /lib/modules/`uname -r` ; pwd  
/lib/modules/2.6.91.6_FC2
```

Las comillas inversas pueden ser usadas dentro de comillas dobles

```
✓ echo "Usted está usando Linux `uname r`"  
Usted está usando Linux 2.6.91.6_FC2
```

✓ `time <comando>`

`time instalador`

`real 0m2.304s` (Tiempo real consumido)

`user 0m0.449s` (Tiempo CPU corriendo el código del programa)

`sys 0m0.106s` (Tiempo CPU corriendo llamadas al sistema)

$\text{real} = \text{user} + \text{espera}$

$\text{espera} = \text{espera de E/S} + \text{tiempo idle (ejecución de otras tareas)}$

- EL shell le permite al usuario definir variables, las cuales pueden ser empleadas en los comandos. La convención es emplear nombres en minúscula.
- Se pueden definir variables de ambiente, que son variables que pueden ser visibles a los scripts o a ejecutables que son llamados desde el shell. La convención es el uso de nombres en mayúscula.

- **env**

Muestra todas las variables de ambiente y su valor.

- Variables del shell (bash)

```
proyecto=/home/maguilar/proyecto/  
ls -la $proyecto ; cd $proyecto
```

- Variables de ambiente

```
cd $HOME  
export DEBUG=1  
./instalador
```

Muestra la información de depuración.

✓ **LD_LIBRARY_PATH**

Directorio de las bibliotecas compartidas.

✓ **DISPLAY**

Id de la pantalla.

✓ **EDITOR**

Editor por defecto (vi,mcedit,...).

✓ **HOME**

Directorio home del usuario actual.

✓ **HOSTNAME**

Nombre de la máquina local

✓ **MANPATH**

Directorios de las páginas de manual.

✓ **PATH**

Directorios de los comandos de shell.

✓ **PRINTER**

Nombre de la impresora por defecto.

✓ **SHELL**

Nombre del shell actual.

✓ **TERM**

Tipo de terminal actual.

✓ **USER**

Nombre del usuario actual.

✓ PATH

Especifica el orden de búsqueda de los comandos de shell.

```
/home/maguilar/bin:/usr/local/bin:  
/usr/bin:/bin:/usr/X11R6/bin:/bin:/usr/bin
```

✓ LD_LIBRARY_PATH

Especifica el orden de búsqueda de las bibliotecas compartidas para ld.

```
/usr/local/lib:/usr/lib:/lib:/usr/X11R6/lib
```

✓ MANPATH

Especifica el orden de búsqueda de los directorios para las páginas de manuales.

```
/usr/local/man:/usr/share/man
```


Los shells permiten definir alias a los comandos, para efectos de ahorrar tiempo al emplear comandos de uso frecuente.

Ejemplos:

✓ `alias ls='ls -la'`

Útil para definir los argumentos por defecto de los comandos.

✓ `alias rm='rm -i'`

Útil para preguntar por confirmación cada vez que se borra un archivo.

✓ `alias fins='find -name instalador'`

Útil para reemplazar comandos largos.

✓ `alias instalador='. /home/maguilar/instalador.sh'`

Útil para agregar un comando al ambiente de ejecución.

(. es un comando del shell para ejecutar el contenido de un script)

Antes de ejecutar un comando, which dice donde se encuentra y cual alias tiene.

✓ `which ls`

```
Alias ls='ls -color=tty'  
      /bin/ls
```

✓ `which alias`

```
/usr/bin/which: no alias in  
(/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin)
```

✓ ~/.bashrc

Es un script de shell que se lee cada vez que una consola de bash es iniciada. Se pueden definir los siguientes aspectos en dicho archivo:

- ✓ Las variables de ambiente por defecto.
- ✓ Los alias.
- ✓ El prompt.
- ✓ Un mensaje de bienvenida.

- Se puede usar las flechas izquierda y derecha para mover el cursor en la línea de comandos actual.
- Se puede usar [CTRL][a] para ir al inicio de la línea y [CTRL][e] para ir al final de la misma.
- Se puede usar las flechas arriba y abajo para seleccionar un comando entre la lista de comandos ejecutados en la consola actual.
- Se puede emplear [TAB] para autocompletar comandos y rutas de directorios para evitar escribir los nombres completos.
- Se puede usar [CTRL][r] para buscar entre un comando específico entre la historia de los comandos ejecutados mediante una palabra clave.

✓ **history**

Muestra los últimos comandos ejecutados y su número. Es posible copiarlos y pegarlos para ejecutarlos nuevamente.

✓ **!!**

Ejecuta el último comando.

✓ **!1003**

Ejecuta el comando por su número.

✓ **!cat**

Ejecuta el último comando que coincida con el patrón.

COMANDOS MISCELÁNEOS

Editores de texto gráficos:

- ✓ nedit
- ✓ Emacs

Editores de texto de consola:

- ✓ vi
- ✓ nano
- ✓ mcedit

<http://www.nedit.org/>

- Es el mejor para usuarios no expertos en vi o emacs.
- Características:
 - ✓ Selección y movimiento de texto sencilla.
 - ✓ Resalta sintáxis para la mayoría de lenguajes y formatos.
 - ✓ Fácil de personalizar mediante los menús.

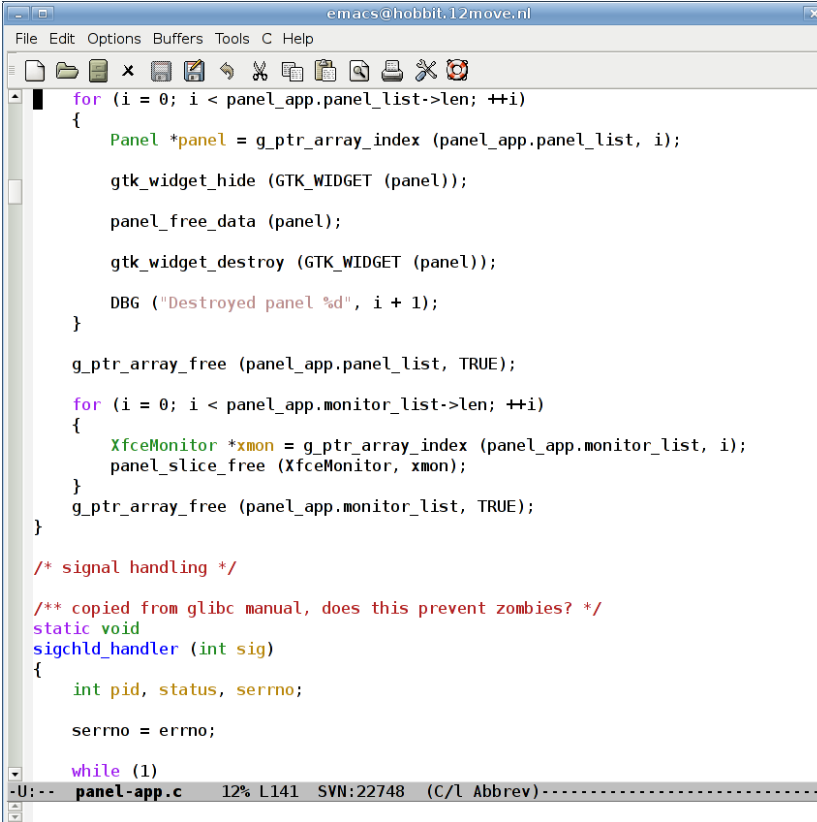
```

nedit.c
File Edit Search Preferences Shell Macro Windows Help
Find: [ ] Rev Literal Case RegEx
/home/schlagtr/SRC/Editors/nedit-5.1beta/source/nedit.c byte 11632, col 8, 21969 bytes

240 };
241
242 static char cmdLineHelp[] =
243 #ifndef VMS
244 "Usage: nedit [-read] [-create] [-line n | +n] [-server] [-do command]\n\
245 [-tags file] [-tabs n] [-wrap] [-nowrap] [-autoindent]\n\
246 [-noautoindent] [-autosave] [-noautosave] [-lm language]\n\
247 [-rows n] [-columns n] [-font font] [-geometry geometry]\n\
248 [-iconic] [-noiconic] [-display [host]:server.screen]\n\
249 [-svrname name] [-xrm resourcestring] [-import file] [file...]\n"
250 #else
251 ""
252 #endif /*VMS*/
253
254 int main(int argc, char **argv)
255 {
256     int i, lineNumber, nRead, fileSpecified = FALSE, editFlags = CREATE;
257     int isServer = FALSE, gotoLine = FALSE, macroFileRead = FALSE;
258     int iconic = FALSE;
259     char *toDoCommand = NULL, *geometry = NULL, *langMode = NULL;
260     char filename[MAXPATHLEN], pathname[MAXPATHLEN];
261     XrmDatabase prefDB;
262     static char *protectedKeywords[] = {"-iconic", "-icon", "-geometry", "-g",
263     "-rv", "-reverse", "-bd", "-bordercolor", "-borderwidth", "-bw",
264     "-title", NULL};
265
266     /* Save the command which was used to invoke nedit for restart command */
267     ArgV0 = argv[0];
268
269     #ifndef NO_XMIM
270     /* Set local for C library and X, and Motif input functions */

```


- ✓ Es un editor de texto muy poderoso.
- ✓ Ideal para usuarios avanzados.
- ✓ Menos ergonómico que nedit.
- ✓ Es mucho más que un editor de texto: juegos, e-mail, shell, browser.



The screenshot shows the Emacs text editor window titled 'emacs@hobbit.12move.nl'. The menu bar includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'C', and 'Help'. The toolbar contains various icons for file operations and editing. The main text area displays C code from 'panel-app.c'. The code includes loops for iterating over 'panel_list' and 'monitor_list', functions for hiding and destroying GTK widgets, and a signal handler 'sigchld_handler'. The status bar at the bottom shows the file path '-U:-- panel-app.c', the cursor position '12% L141', and version information 'SVN:22748 (C/l Abbrev)'.

```
for (i = 0; i < panel_app.panel_list->len; ++i)
{
    Panel *panel = g_ptr_array_index (panel_app.panel_list, i);

    gtk_widget_hide (GTK_WIDGET (panel));

    panel_free_data (panel);

    gtk_widget_destroy (GTK_WIDGET (panel));

    DBG ("Destroyed panel %d", i + 1);
}

g_ptr_array_free (panel_app.panel_list, TRUE);

for (i = 0; i < panel_app.monitor_list->len; ++i)
{
    XfceMonitor *xmon = g_ptr_array_index (panel_app.monitor_list, i);
    panel_slice_free (XfceMonitor, xmon);
}

g_ptr_array_free (panel_app.monitor_list, TRUE);
}

/* signal handling */

/** copied from glibc manual, does this prevent zombies? */
static void
sigchld_handler (int sig)
{
    int pid, status, serrno;

    serrno = errno;

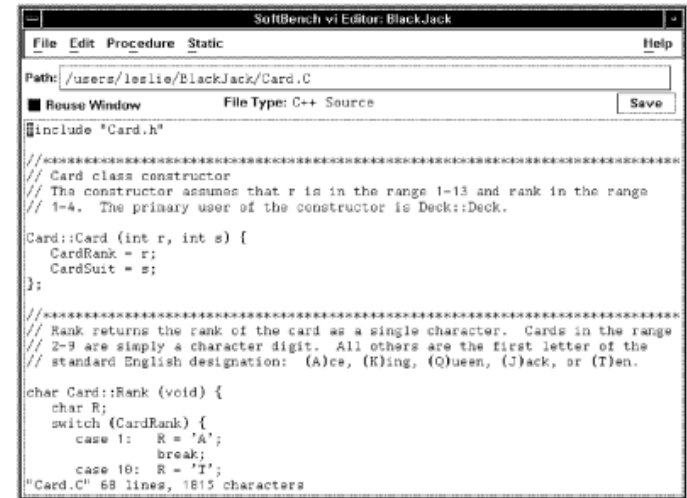
    while (1)
```

Es un editor de texto disponible en la mayoría de sistemas UNIX. Fue creado antes de que las computadoras con mouse aparecieran.

✓ Difícil de aprender para usuarios principiantes acostumbrados a editoriales de texto gráficos.

✓ Muy productivo para usuarios avanzados.

✓ Muy empleado en administración de sistemas y sistemas empujados, donde solo se tiene una consola de texto.



```
SoftBench vi Editor: BlackJack
File Edit Procedure Static Help
Path: /users/leslie/BlackJack/Card.C
Reuse Window File Type: C++ Source Save

#include "Card.h"

//*****
// Card class constructor
// The constructor assumes that r is in the range 1-13 and rank in the range
// 1-4. The primary user of the constructor is Deck::Deck.

Card::Card (int r, int s) {
    CardRank = r;
    CardSuit = s;
};

//*****
// Rank returns the rank of the card as a single character. Cards in the range
// 2-9 are simply a character digit. All others are the first letter of the
// standard English designation: (A)ce, (K)ing, (Q)ueen, (J)ack, or (T)en.

char Card::Rank (void) {
    char R;
    switch (CardRank) {
        case 1: R = 'A';
                break;
        case 10: R = 'T';
    }
}

"Card.C" 68 lines, 1815 characters
```

<http://www.nanoeditor.org/>

- ✓ Es un editor de consola.
- ✓ Amigable y fácil de aprender para usuarios principiantes.
- ✓ Es una alternativa a vi en sistemas empuotrados.
- ✓ Disponible en muchas plataformas.



The screenshot shows the GNU nano 0.9.99pre3 text editor interface. The title bar indicates the file being edited is 'nano.c'. The main window displays the source code of the nano editor, including copyright information for Chris Allegretta (1999) and the GNU General Public License. The status bar at the bottom shows various keyboard shortcuts for navigation and editing, such as 'G Get Help', 'X Exit', 'W WriteOut', 'R Read File', 'N Replace', 'U Where Is', 'Y Prev Page', 'O Next Page', 'K Cut Text', 'U UnCut Text', 'C Cur Pos', and 'T To Spell'. A message 'Read 2583 lines' is also visible in the status bar.

- ✓ Es un editor de consola.
- ✓ Forma parte del paquete Midnight Commander (mc) .
- ✓ Es fácil de emplear para usuarios principiantes.
- ✓ Permite resaltar la sintaxis para ficheros de código fuente de ciertos lenguajes de programación.
- ✓ Tiene la capacidad de trabajar tanto en modo ASCII como en modo Hexadecimal.



```
mc - ~/sc_tutorial/simpleCounterSC
sc_main.cpp  [-H-]  0 L:  2+ 0  2/100] *(22 /1957b)= . 10 0x0A

// Counter module
class counter: public sc_module {
    int value;
public:
    sc_in<bool> clk;
    sc_in<bool> count;
    sc_in<bool> reset;
    sc_out<int> q;

    SC_HAS_PROCESS(counter);

    counter(sc_module_name nm): sc_module(nm), value(0) {
        SC_METHOD(do_count);
        sensitive<< clk.pos() << reset ;
    }
protected:
    void do_count() {
        if (reset ) { value = 0; }
        else if (count) {
            value++;
            q.write(value);
        }
    }
};

// Testbench
class testbench: public sc_module {
public:
    sc_out<bool> clk;
    sc_out<bool> reset;
    sc_out<bool> count;
    SC_HAS_PROCESS(testbench);
    testbench(sc_module_name nm): sc_module(nm) {
        SC_THREAD(clk_gen);
        SC_THREAD(stimuli);
    }
    void clk_gen() {
        while(true) {
            clk.write(true);
            wait(10, SC_NS);
        }
    }
};
```

✓ **du -h <archivo>** (uso de disco, diferente al tamaño del archivo)

-h: retorna el tamaño en disco de un archivo dado, en : K (kilobytes), M (megabytes) o G (gigabytes). Sin **-h**, **du** retorna el número crudo de bloques de disco empleados por el archivo.

✓ **dh -sh <directorio>**

-s: retorna la suma del uso del disco de todos los archivos en el directorio dado.

✓ **df -h <directorio>**

Retorna el uso del disco y el espacio libre para el sistema de archivos que contiene el directorio dado.

Ejemplo:

df -h .

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/hda5	9.2G	7.1G	1.8G	81%	/

✓ **df -h**

Retorna la información del espacio de disco para todos los sistemas de archivos en el sistema.

✓ `g[un]zip <archivo>`

Es la aplicación GNU para compresión. Crea archivo `.gz`, con un rendimiento similar a Zip.

✓ `b[un]zip2 <archivo>`

Más reciente y efectiva utilidad de compresión. Crea archivos `.bz2`. Es un 20-25% mejor que `gzip`.

✓ `7-zip`

Presenta una mejor compresión que bzip2, entre un 10 y un 20%.

Útil para respaldar y liberar un conjunto de archivos en uno solo.

✓ **tar** (tape archive)

✓ **tar cvf <archivero> <archivos o directorios>**

Crea un archivo.

c: crear

v: verbose, muestra el proceso de archivamiento.

f: archivero creado en un archivo.

Ejemplo:

```
tar cvf /backup/home.tar /home
```

```
bzip2 /backup/home.tar
```


✓ **tar tvf <archivero>**

Ver los contenidos o verificar integridad.

t: test

✓ **tar xvf <archivero>**

Extraer todos los archivos.

✓ **tar xvf <archivero> <archivos o directorios>**

Los archivos y directorios están dados con rutas relativas a la raíz del archivero.

Un solución de bajo costo para verificar la integridad de los archivos.

✓ **md5sum instalador*.iso > MD5SUM**

Calcula un checksum MD5 (Message Digest Algorithm 5) de 128 bits de los archivos dados.

Ejemplo:

```
db8c7254beeb4f6b891d1ed3f689b412 instalador1.iso
2c11674cf429fe570445afd9d5ff564e instalador2.iso
f88f6ab5947ca41f3cf31db04487279b instalador3.iso
6331c00aa3e8c088cc365eeb7ef230ea instalador4.iso
```

✓ **md5sum -c MD5SUM**

Chequea la integridad de los archivos en MD5SUM comparando MD5 checksum actual con el original.

- ***Multi-usuario, multi-tarea, multi-cliente, multi-impresora:*** En UNIX / Linux, los comandos de impresión no imprimen realmente. Ellos envían tareas a las colas de impresión, posiblemente en la máquina local o a través de una red.
- ***Sistema de impresión independiente de la impresora:*** Los servidores de impresión aceptan trabajos en PostScript o texto. Los controladores (drivers) de las impresoras en los servidores son los que se preocupan por la conversión de los datos al formato propio de cada impresora.
- ***Sistema robusto:*** aunque se reinicie un sistema los trabajos de impresión quedarán pendientes.

PRINTER, define la impresora por defecto del sistema. Ejemplo: **export PRINTER=lp**.

✓ **lpr [-P<cola>] <archivos>**

Envía los archivos dados a la cola de impresión especificada. Los archivos deben ser texto o el formato PostScript, de otra manera solo se imprimirá basura.

✓ **a2ps [-P<cola>] <archivos>**

“Any to PostScript” convierte muchos formatos a PostScript y envía la salida a la cola de impresión.

✓ `lpq [-P<cola>]`

Lista todos los trabajos de impresión en la cola dada o la cola por defecto.

`lp is not ready`

Rank	Owner	Job	File(s)	Total Size
1st	maguilar	84	cv.ps	60416 bytes
2nd	maguilar	85	tarea.ps	65024000 bytes

✓ `cancel <trabajo#> [<cola>]`

Remueve el trabajo con el número dado de la cola por defecto.

Para ver un archivo PostScript

- Los visualizadores de PostScript existen pero su calidad es pobre.
- Es mejor convertirlos a PDF con **ps2pdf**:

```
ps2pdf tarea.ps
```

```
xpdf tarea.pdf &
```

Para imprimir un archivo PDF

- No se necesita abrir un lector de PDFs.
- Es mejor convertirlo a PostScript con **pdf2ps**:

```
pdf2ps tarea.ps
```

```
lpr tarea.ps
```

✓ `diff <archivo1> <archivo2>`

Reporta diferencias entre dos archivos o nada si son idénticos.

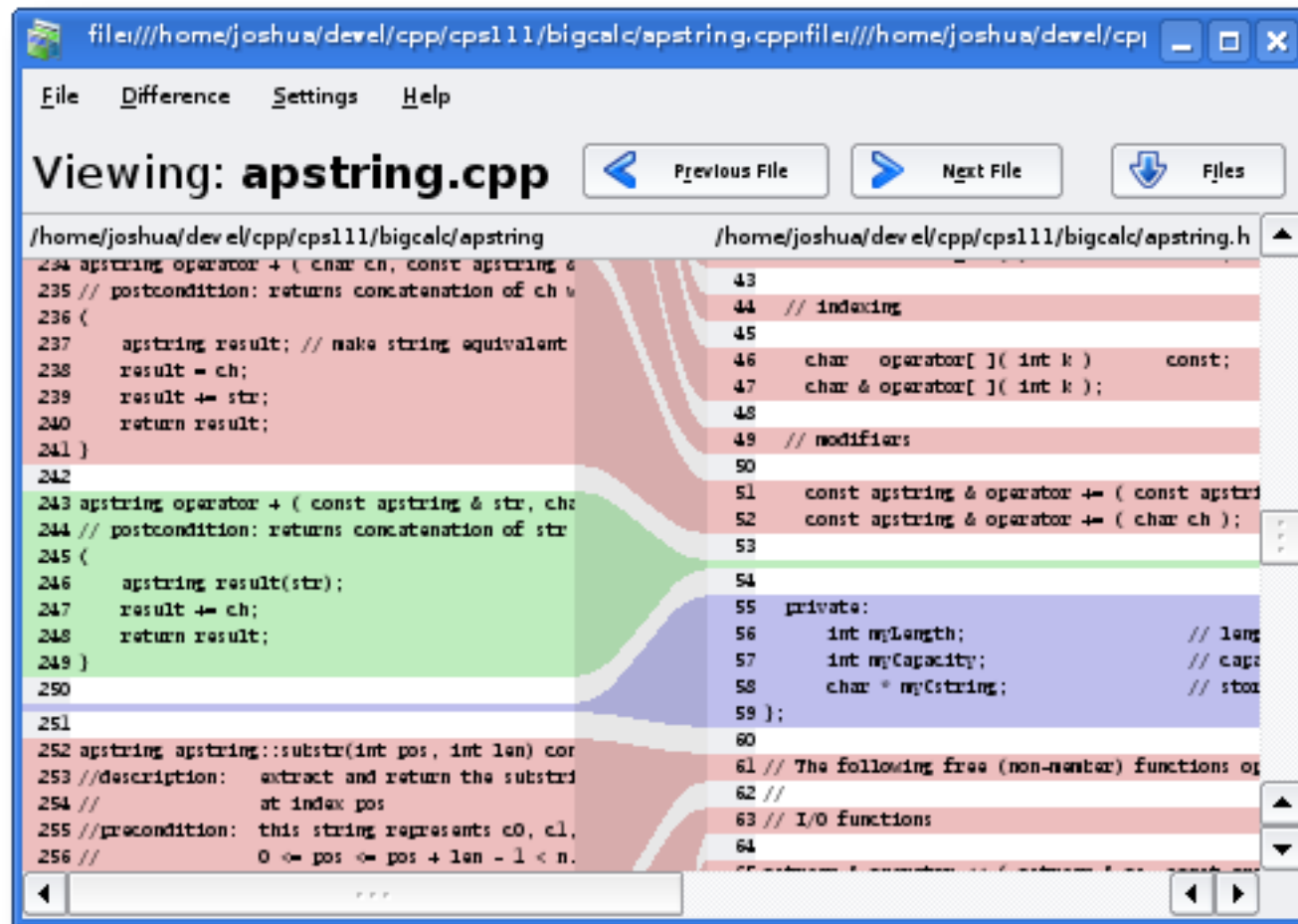
✓ `diff -r <directorio1> <directorio2>`

Reporta todas las diferencias entre archivos con el mismo nombre en los dos directorios.

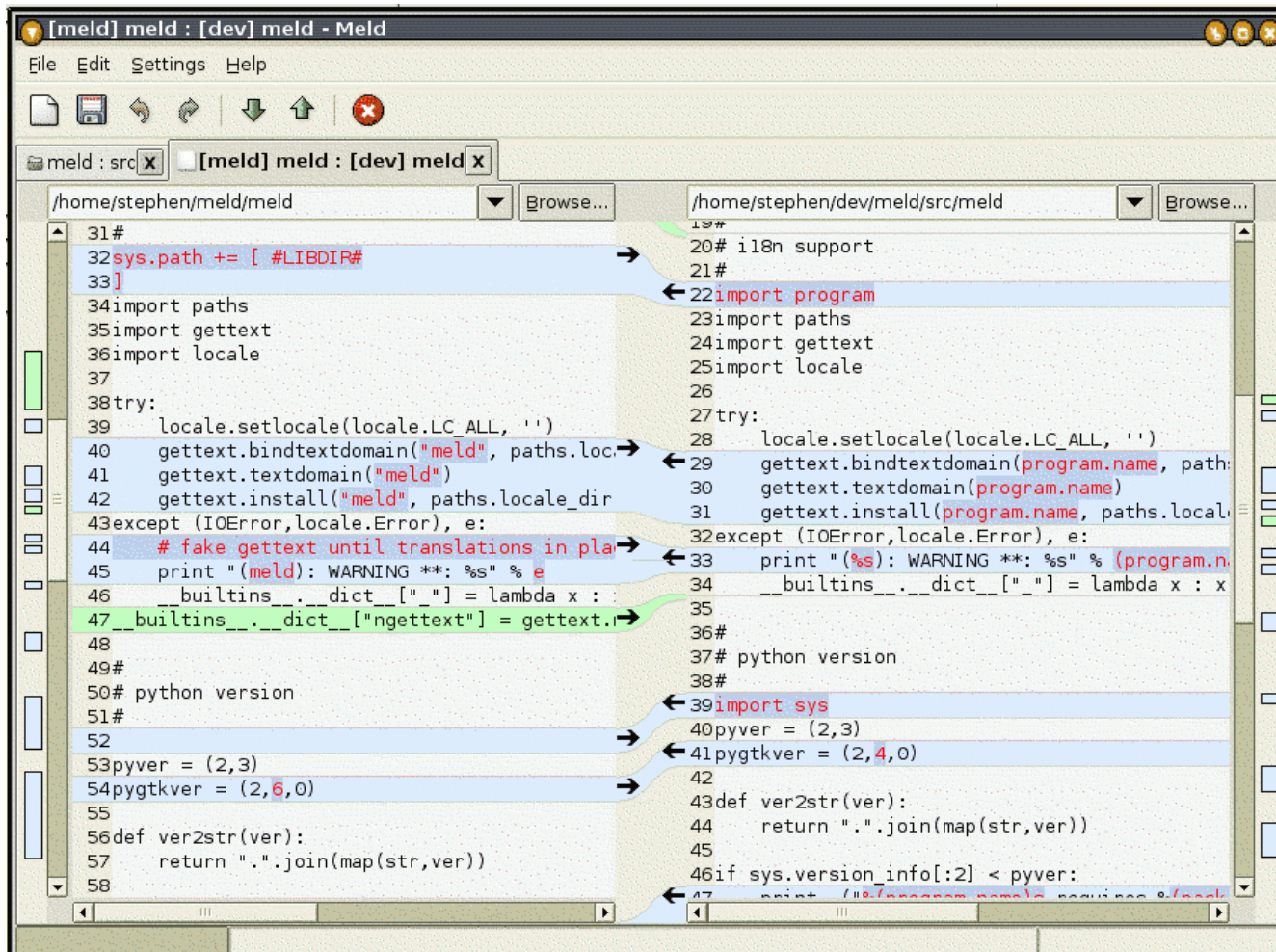
La herramienta `diff` es empleada para generar parches de software.

Para investigar las diferencias en detalle, es mejor usar herramientas gráficas.

Es una herramienta gráfica muy útil para compara archivos y mezclar diferencias entre archivos.



Es una de las herramientas gráficas de comparación de archivos más poderosas. Se puede comparar dos o inclusive tres archivos.



✓ `find . -name "*.pdf"`

Lista todos los archivos *.pdf en el directorio actual o subdirectorios. Las comillas dobles son necesarias para evitar que el shell expanda el caracter *.

✓ `find docs name "*.pdf" exec xpdf {} ';'``

Encuentra todos los archivos *.pdf en el directorio docs y despliega uno tras otro.

Es una alternativa de búsqueda basada en expresiones regulares.

✓ `locate llave`

Muestra todos los archivos en el sistema con llave en sus nombres.

✓ `locate "*.pdf"`

Muestra todos los archivos `*.pdf` disponibles en la máquina completa.

✓ `locate "/home/maguilar/*leccion*"`

Muestra todos los archivos `*leccion*` en el directorio dado.

`locate` es más rápido porque indexa todos los archivos en una basa de datos dedicada, que es actualizada regularmente.

`find` es una mejor utilizada de búsqueda para archivo recientemente creados.

En vez de descargar los archivos mediante un browser, se puede solo copiar el URL en la consola y descargarlo con **wget**.

- ✓ Soporte de http y ftp.
- ✓ Puede continuar desde el mismo descargas interrumpidas.
- ✓ Puede descargar sitios enteros.
- ✓ Es muy útil en scripts.

Ejemplos:

✓ **wget http://www.descargas.com/instalador.iso**

Descarga un archivo, si se agrega **-c** continua una descarga interrumpida.

✓ **wget -m http://www.descargas.com**

Hace un mirror de un sitio.

✓ **wget http://www.descargas.com/libro/**

Hace descargas recursivas de un libro en línea para un acceso fuera de línea.

-np: “no parent”. Sigue solo enlaces en el directorio actual.

✓ **sleep 60**

Espera 60 segundos.

✓ **wc reporte.txt** (word count)

438 2115 18302 reporte.txt

Cuenta el número de líneas, palabras y caracteres del archivo dado o de la entrada estándar.

✓ **bc (basic calculator)**

Es un calculadora completa.

✓ **date**

Retorna la fecha actual del sistema.

BASES DE LA ADMINISTRACIÓN DEL SISTEMA

- Los privilegios de root son solo necesitados por tareas específicas con riesgos de seguridad: montaje de sistemas de archivos, crear archivos de dispositivos, carga de drivers, inicio de redes, actualización de paquetes, etc.
- Inclusive si se tiene el password de root, las cuentas regulares deben ser el 99.9% suficientes para realizar las tareas, a menos de que sea un administrados del sistema.
- En la vida real, probablemente no se tenga acceso a la cuenta de root.

- En caso de que realmente se quiera usar **root**...

✓ **su - (switch user)**

Si tiene el password de **root**:

En distribuciones modernas, el comando **sudo** le da acceso a algunos privilegios de **root** con el password común del usuario.

Ejemplo:

```
sudo mount /dev/hda4 /home
```


✓ `useradd <usuario>`

Crea una cuenta para el `<usuario>`.

✓ `passwd <usuario>`

Crea una contraseña para el usuario. También puede ser empleada para cambiar la contraseña de un usuario existente.

✓ `userdel -r <usuario>`

Elimina al `<usuario>`, junto con todo su directorio home (`-r`).

✓ `groupadd <grupo>`

Crea un `<grupo>`.

✓ `groupdel <grupo>`

Elimina al `<grupo>`.

✓ `gpasswd -a <usuario> <grupo>`

Asigna el `<usuario>` al `<grupo>`.

✓ `who`

Muestra todos los usuarios registrados en el sistema.

✓ `whoami`

Muestra como está registrado el usuario actual.

✓ `groups`

Muestra a cuales grupos pertenece el usuario actual.

✓ `groups <usuario>`

Muestra a cuales grupos pertenece `<usuario>`.

✓ `finger <usuario>`

Muestra más detalles acerca del `<usuario>` (nombre real, etc),

No es necesario cerrar la sección de un usuario para abrir la de otro usuario.

✓ **su maguilar**

Cambia al usuario **maguilar**, pero mantiene las variables de ambiente del usuario original.

✓ **su - maguilar**

Se registra como maguilar con la configuración de usuario nuevo.

✓ **su -**

Cuando no se da argumentos, significa usuario **root**.

✓ **ifconfig -a**

Muestra los detalles de todas las interfaces de red disponibles en el sistema.

✓ **ifconfig eth0**

Muestra los detalles de la interface de red eth0.

✓ **ifconfig eth0 192.168.1.100**

Asigna el IP dado a eth0.

✓ **ifconfig eth0 down**

Apaga la interface de red eth0.

✓ `route add default gw 192.168.0.1`

Define la ruta de los paquetes fuera de la red local. El gateway (`192.168.0.1`) es el responsable de enviar los paquetes al siguiente gateway, hasta el destino final.

✓ `route -n`

Lista las rutas existentes.

✓ `route del default` ○ `route del <ip>`

Elimina la ruta dada.

✓ `ping www.google.com` o `ping 192.168.1.1`

Intenta enviar paquetes a la dirección indicada y obtiene el reconocimiento de los paquetes de retorno.

```
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.  
64 bytes from 192.168.1.1: icmp_seq=0 ttl=150 time=2.51 ms  
64 bytes from 192.168.1.1: icmp_seq=1 ttl=150 time=3.16 ms  
64 bytes from 192.168.1.1: icmp_seq=2 ttl=150 time=2.71 ms  
64 bytes from 192.168.1.1: icmp_seq=3 ttl=150 time=2.67 ms
```

- Cuando se puede hacer ping hacia el gateway, la interface de red funciona bien.
- Cuando se puede hacer ping a un ip externo, la configuración de la red está correcta.

1. Conectese a la red (cable o inalámbrica).

2. Identifique su interface de red.

```
ifconfig a
```

3. Asigne un ip a su interface.

```
ifconfig eth0 192.168.0.100
```

4. Agregue una ruta a su gateway para los paquetes fuera de la red.

```
route add default gw 192.168.0.1
```


Los programas necesitan saber que dirección IP corresponde el nombre dado, por ejemplo `www.google.com`.

Los Domain Name Server (DNS) son los que se encargan de esto.

Nada más se tiene que especificar la dirección IP de uno o más servidores DNS en el archivo `/etc/resolv.conf`:

```
nameserver 217.19.192.132
```

```
nameserver 212.27.32.177
```

Los cambios son inmediatos.

✓ `mkfs.ext2 /dev/sda1`

Formatea una llave USB en el formato ext2.

✓ `mkfs.ext2 -F disco.img`

Formatea un archivo de imagen de disco en el formato ext2.

✓ `mkfs.vfat -v -F 32 /dev/sda1 (-v:verbose)`

Formatea una llave USB en el formato FAT32.

✓ `mkfs.vfat -v -F 32 disco.img`

Formatea un archivo de imagen de disco en FAT32.

Las imágenes de archivos en blanco pueden ser creadas de la siguiente manera:

`dd if=/dev/zero of=disco.img bs=1024 count=65536`

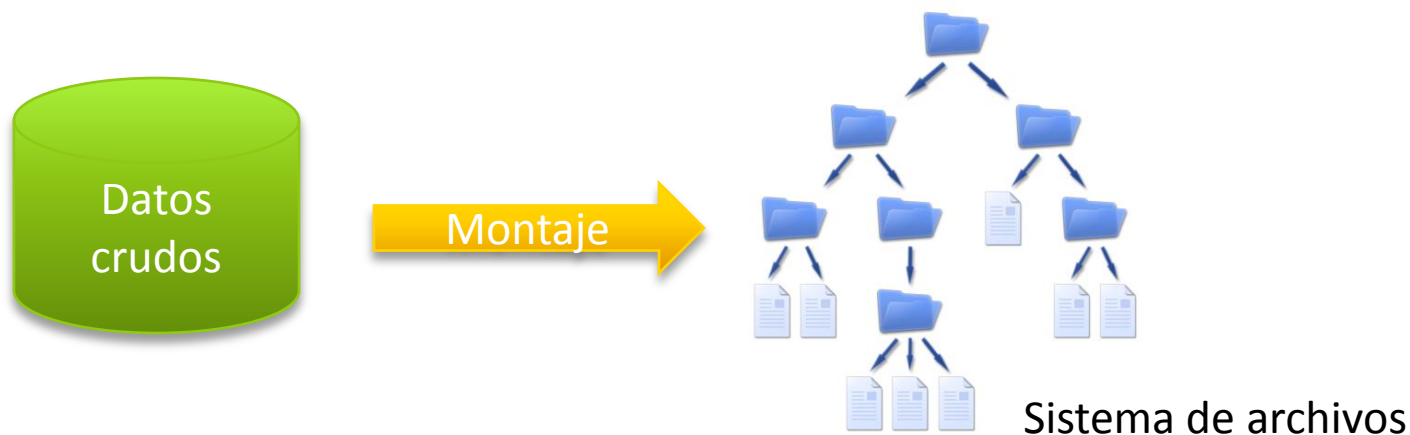
- Para usar sistemas de archivos en cualquier dispositivo visible en un sistema se deben montar.

- Lo primero es crear un punto de montaje en el sistema.

```
mkdir /mnt/discoUSB
```

- Lo segundo es montarlo.

```
mount -t vfat /dev/sda1 /mnt/usbdisk
```



- Hay muchas opciones disponibles en mount, en particular para escoger los permisos, los propietarios de los archivos y el grupo, revise la página de manual de mount para más detalles.
- La información de los dispositivos montados en un sistema se almacenan en el archivo /etc/fstab

```
# /etc/fstab: static file system information.
```

# <file system>	<mount point>	<type>	<options>	<dump>	<pass>
proc	/proc	proc	defaults	0	0
/dev/hda3	/	ext3	defaults	0	1
/dev/hda4	/home	ext3	defaults	0	2
/dev/hda2	/root2	ext3	defaults	0	2
/dev/hda1	none	swap	sw	0	0
/dev/hdc	/media/cdrom0	udf,iso9660	user,noauto	0	0

Es también posible montar sistemas de archivos almacenados en archivos regulares.

- ✓ Es útil para desarrollar sistemas de archivos para otra máquina.
- ✓ Es útil para tener acceso a imágenes ISO sin necesidad de quemar un CD.
- ✓ Es útil para tener un sistema de archivos en Linux dentro de un archivo en una partición de Windows.

```
cp /dev/sda1 usbkey.img
```

```
mount -o loop -t vfat usbkey.img /mnt/usbdisk
```

Con solo el comando mount sin argumentos:

```
/dev/hda6 on / type ext3 (rw,noatime)
none on /proc type proc (rw,noatime)
none on /sys type sysfs (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/hda4 on /data type ext3 (rw,noatime)
none on /dev/shm type tmpfs (rw)
/dev/hda1 on /win type vfat (rw,uid=501,gid=501)
none on /proc/sys/fs/binfmt_misc type binfmt_misc
(rw)
```

✓ `umount /mnt/discoUSB`

Realiza todas las escrituras pendientes y desmonta el dispositivo dado, para que puede ser removido de una forma segura.

- Para poder desmontar un dispositivo, se tiene que cerrar todos los archivos abiertos de el:

- ✓ Cerrar todas las aplicaciones que tenga dados abiertos en la partición.

- ✓ Asegurarse que ninguna shell tiene un directorio actual en el punto de montaje del dispositivo.

- ✓ Para verificar si hay procesos que tiene abiertos archivos en la partición se puede usar `fuser -mv <mount point>`. También se puede usar `k` para matar dichos procesos.

✓ **halt**

Detiene el sistema inmediatamente.

✓ **reboot**

Reinicia el sistema inmediatamente.

Las siguientes son instrucciones para paquetes basados en Debian GNU/Linux.

✓ `/etc/apt/sources.list`

Lista de repositorios de paquetes.

✓ `sudo apt-get update`

Para actualizar la lista de paquetes de los repositorios.

✓ `sudo apt-cache search <palabra_clave>`

Se usa para encontrar el nombre de un paquete a ser instalado.

✓ **sudo apt-get install <paquete>**

Para instalar el paquete dado.

✓ **sudo apt-get remove <paquete>**

Para desinstalar el paquete dado.

✓ **sudo apt-get upgrade**

Para actualizar los paquetes instalados en el sistema.

Presenta la descripción, versión, fuentes, etc de los paquetes.

✓ **dpkg -s <paquete>**

Para distribuciones basadas en Debian (Debian, Ubuntu...).

✓ **rpm -qi <paquete>**

Para distribuciones basadas en RPM (Red Hat, Fedora, Mandriva, Suse...).

SSH

- SSH – stands for Secure Shell.
- SSH es un protocolo de comunicación segura que permite registrarse remotamente, transferencia de archivos remotos, etc.
- Es el reemplazo para telnet, rlogin, rsh, etc.
- En Linux la principal implementación es OpenSSH, tanto para el cliente como servidor.

- OpenSSH está disponible en todas las distribuciones GNU/Linux.
- En Ubuntu, los paquetes son:
 - ✓ openssh-client
 - ✓ openssh-server
- Conectarse a un servidor SSH, es tan simple como:
ssh usuario@host
- SSH mostrará un mensaje para ingresar el password y así registrarse en el sistema remoto.

✓ `scp <archivo> usuario@host:<ruta>`

Los archivos pueden ser transferidos remotamente usando scp:

Ejemplo:

```
scp tarea.txt maguilar@192.168.1.40:/home/maguilar/
```

- Con la opción `-x`, se habilita el redireccionamiento de X11, esto permite lo siguiente:

- ✓ Permite a las aplicaciones gráficas que residen en el servidor ser desplegadas en la máquina local.

- ✓ En el servidor, X11Forwarding debe estar habilitado en el archivo de configuración `/etc/ssh/sshd_config`.

✓ `ssh usuario@host ls`

Es útil en scripts de shell.

- SSH es usado por otros programas como capa de transporte como es el caso de rsync, cvs, subversion, etc.

Ejemplo:

`rsync -e ssh ~/trabajo usuario@host:~/trabajo`

- Una característica interesante de SSH es que se puede evitar el ingreso del password mediante el uso de llaves criptográficas. Los pasos para hacer esto son los siguientes:

✓ `ssh-keygen`

Crea una llave pública y privada. Pedirá un passphrase, que se requiere para desbloquear la llave privada cada vez que se use, por comodidad se puede dejar en blanco.

La llave es generada en:

`~/.ssh/id_rsa`, es la llave privada y nadie debe tener acceso a ella solo el usuario propietario.

`~/.ssh/id_rsa.pub`, es la llave pública y puede ser transferida a cualquiera.

✓ `ssh-copy-id usuario@host`

Con esto se copia la llave pública local al host remoto que se desea conectar. La llave pública se almacena en `~/.ssh/authorized_keys` en el host remoto.

Cada vez que se desea registrar al host remoto solo preguntará por el passphrase en caso de que se haya digitado alguno.

✓ `ssh-agent`

Esto evita estar digitando el passphrase cada vez que se registra al host, debido a que lo guarda en memoria.

✓ `ssh-add`

Le da el passphrase al agente.