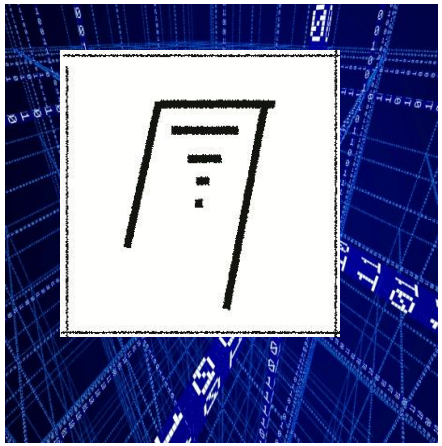
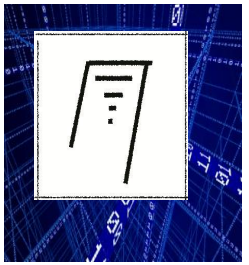


Sistemas Operativos – Linux Básico

¿Qué es un Sistema Operativo (SO)?

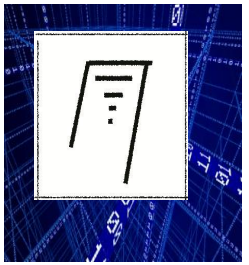


- Es el programa más importante que corre sobre la computadora. Cualquier computadora de propósito general debe tener un SO para poder ejecutar programas. Los SO realizan tareas básicas como reconocimiento de la entrada de datos desde el teclado, enviar datos a la pantalla, la administración de los archivos y directorios almacenados en las unidades de disco duro y el control de los dispositivos periféricos como impresoras, scanners, unidades de almacenamiento externas, entre otros.

- Para sistemas muy grandes tienen todavía más responsabilidades y tareas. Su labor es como la de un policía de tránsito, se asegura que todos los programas y usuarios obtengan los recursos que necesitan e interactúen sin que unos intervengan con las actividades de otros.

- El SO también es responsable de la seguridad, se asegura de que usuarios no autorizados no accesen al sistema.

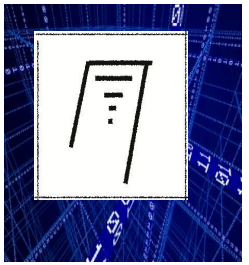
Clasificación de los SO’s



1) **Multiusuario:** Permite que dos o más usuarios ejecuten programas al mismo tiempo. Varios SO's permiten cientos o hasta miles de usuarios.
2) **Multiprocesamiento**: Soporta la ejecución de un mismo programa en más de un procesador.
3) **Multitarea**: Permite que más de un programa se ejecute a la vez.
4) **Multihilo**: Permite que diferentes partes de un mismo programa se ejecuten al mismo tiempo.
5) **Tiempo real**: Responde a los datos de entrada instantáneamente.

- Los SO proveen una plataforma para que otros programas (llamados aplicaciones) puedan ejecutarse. La elección del sistema operativo determina las aplicaciones que pueden ejecutarse en cierto momento.

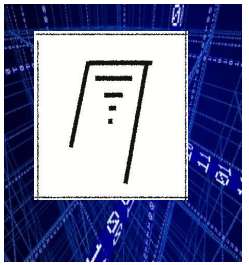
Interfaz con el usuario



- Como usuario uno normalmente interactúa con el SO a través de un conjunto de comandos, por ejemplo en DOS, Linux y Unix se puede crear una carpeta usando el comando **MKDIR** o **MOVE** para mover archivos de un lugar a otro. Los comandos son aceptados y ejecutados por el sistema operativo a través de una aplicación llamada **interprete de línea de comandos**.

- **Las interfaces gráficas (GUI)** permiten la ejecución de los mismos comandos más fácilmente, haciendo clic y ejecutando acciones sobre algunos **menús** (sin necesidad de memorizar los comandos) ejemplos de estos son: las diferentes versiones de Windows (a partir de win'95), las nuevas versiones de Linux como Red Hat y Mandrake o las Mac.

Macintosh - Apple Computer

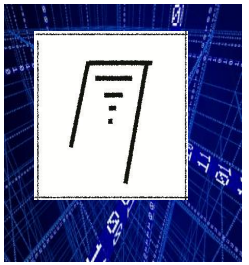


- Es un modelo particular de computadoras hechas por Apple Computer. Fueron introducidas al mercado en 1984, con características como: interfaz gráfica (GUI) usando ventanas, íconos y un ratón para hacerla relativamente fácil para inexpertos y así comenzar a usar la computadora más rápidamente para generar productividad.
- La interfaz gráfica es con la finalidad de evitar el aprendizaje de comandos, de esta forma solamente se apunta sobre el ícono y se hace un clic con el ratón para ejecutar alguna acción.

- La interfaz gráfica está embebida en el sistema operativo, esto significa que todas las aplicaciones que corran sobre una computadora Macintosh tienen una interfaz similar.
- una vez que un usuario se ha familiarizado con una aplicación, aprenderá nuevas aplicaciones más rápidamente.

- El éxito de la GUI de Macintosh fue el inicio de una nueva era de aplicaciones y sistemas operativos basados en interfaces gráficas.
- La interfaz de Windwos copia muchas de las características de la Mac
- Hay muchos modelos diferentes de Macintosh, con diferentes grados de rápidez y poder. Todos los modelos están disponibles en diferentes configuraciones.
- Todos los modelos desde 1994 están basados el microprocesador PowerPC

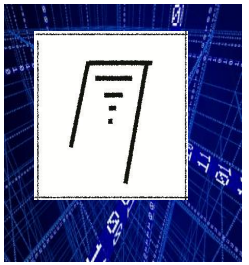
Microsoft Windows



- Es una familia de Sistemas Operativos para computadoras personales. Windows domina el mercado mundial de las computadoras personales, según algunas estimaciones está en un 90% de todas las computadoras personales.

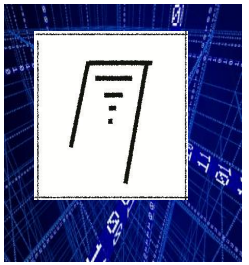
- Del restante 10% la mayoría son computadoras Macintosh. Igual que el entorno de las computadoras Macintosh, Windows provee interfaz gráfica (GUI), administración de memoria virtual, multitareas y soporte para una gran variedad de dispositivos periféricos.

UNIX



- UNIX es uno de los más populares sistemas operativos multiusuarios y multitareas que fue desarrollado por los laboratorios Bell a inicios de los 70's. Fue diseñado para ser pequeño y flexible, siendo éste un sistema operativo usado exclusivamente por programadores.

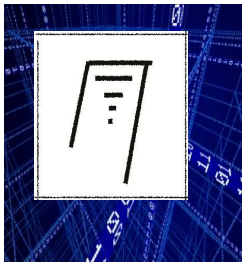
- **UNIX fue uno de los primeros sistemas operativos que fueron escritos en un lenguaje de alto nivel (lenguje C). Esto significa que podía ser instalado en virtualmente cualquier computadora en el que existtiera un compilador de C**. Esta portabilidad natural, combinada a su bajo costo lo convirtió en una de las elecciones más populares entre las universidades. En realidad no era caro porque los laboratorios Bell prohibían su comercio a gran escala. Debido a esta portabilidad Unix se convirtió rápidamente en el sistema operativo líder para los equipos de cómputo científico.



- Los laboratorios Bell distribuían el Sistema Operativo en su código fuente para que cualquiera que obtuviera una copia pudiera modificarlo y acoplarlo para sus propias necesidades. A finales de los 70's hubieron muchas versiones modificadas, esto provocó que se hiciera un estándar debido a la falta de compatibilidad entre las diferentes versiones.

- A finales del siglo pasado fue creado el proyecto Linux por Linus Torvalds a partir de Multics (antecesor de Unix). Linux es, a simple vista, un Sistema Operativo.
- Es una implementación de libre distribución UNIX para computadoras personales (PC), servidores, y estaciones de trabajo.
- Linux Fue desarrollado para el i386 y ahora soporta los procesadores i486, Pentium, Pentium Pro y Pentium II, así como los clones AMD y Cyrix. También soporta máquinas basadas en SPARC, DEC Alpha, PowerPC/PowerMac, y Mac/Amiga Motorola 680x0.

Características de Linux



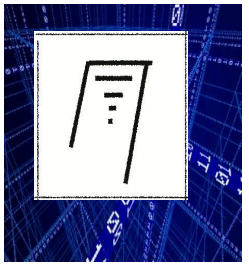
- **Multitarea:** La palabra multitarea describe la habilidad de ejecutar varios programas al mismo tiempo. LINUX utiliza la llamada multitarea preeventiva, la cual asegura que todos los programas que se estan utilizando en un momento dado seran ejecutados, siendo el sistema operativo el encargado de ceder tiempo de microprocesador a cada programa.

- **Multiusuario**: Muchos usuarios usando la misma maquina al mismo tiempo.

- **Multiplataforma**: Las plataformas en las que en un principio se puede utilizar Linux son 386-, 486-. Pentium, Pentium Pro, Pentium II,Amiga y Atari, tambien existen versiones para su utilizacion en otras plataformas, como Alpha, ARM,MIPS, PowerPC y SPARC.

- **Multiprocesador**: Soporte para sistemas con mas de un procesador esta disponible para Intel y SPARC.

- **Protección de la memoria entre procesos**, de manera que uno de ellos no pueda colgar el sistema.



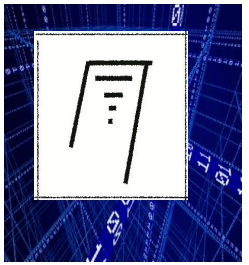
- **Carga de ejecutables por demanda**: Linux sólo lee del disco aquellas partes de un programa que están siendo usadas actualmente.

- **Todo el código fuente está disponible**, incluyendo el núcleo completo y todos los drivers, las herramientas de desarrollo y todos los programas de usuario; además todo ello se puede distribuir libremente. Hay algunos programas comerciales que están siendo ofrecidos para Linux actualmente sin código fuente, pero todo lo que ha sido gratuito sigue siendo gratuito.

- **Soporte para varios sistemas de archivo comunes**, incluyendo minix-1, Xenix y todos los sistemas de archivo típicos de System V, y tiene un avanzado sistema de archivos propio con una capacidad de hasta 4 Tb y nombres de archivos de hasta 255 caracteres de longitud
- **Acceso transparente a particiones MS-DOS y Windows**

- **Sistema de archivos de CD-ROM** que lee todos los formatos estándar de CD-ROM.
- TCP/IP, incluyendo ftp, telnet, NFS, etc.
- Software cliente y servidor Netware.
- Diversos protocolos de red incluidos en el kernel: TCP, IPv4, IPv6, AX.25, X.25, IPX, DDP, Netrom, etc.

Iniciando Linux



- **Para iniciar Linux:**
  - Primero, el sistema pide el login, posteriormente el password (no se ve cuando se teclea) por ejemplo:

```
login as: sanson
sanson@ndikandi.utm.mx's password: _
```

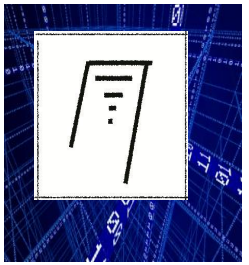
- Lo primero que hay que ver es el **contenido** del sistema, para ello se puede usar el **comando ls**.
  - Ejemplo:

```
bash-2.05$ ls
Enviados                dead.letter             numeros.o
. . .
Papelera                impo.java               Pospuestos
```

- Otra opción es

```
bash-2.05$ ls -al
drwxr-xr-x   8    computo    1024 Nov  3 05:27 .
drwxr-xr-x 379 root  root       6656 Nov  4 05:53 ..
-rwx--x--x   1    computo    1783 Oct 30  2002 .addressbook
-rwx--x--x   1    computo    2456 Oct 30  2002 .addressbook.lu
-rwx--x--x   1    computo    6030 Nov  5 05:24 .bash_history
-rwx--x--x   1    computo    1328 Sep  1  2002 .cshrc
```

Línea de comandos



ls -m ← listar archivos separados por comas.

ls –x ← Muestra archivos y directorios ordenados por columnas antes que por filas

ls –F ← muesra una diagonal en donde hay directorios después del nombre, un * si el archivo es ejecutable.

ls –a ← muestra todos los archivos

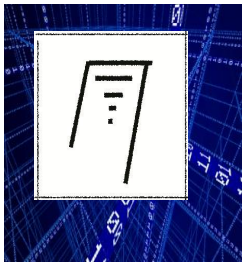
ls –l ← muestra en formato largo (muestra permisos, número de ligas, usuario propietario, grupo al que pertenece, tamaño del archivo, hora de última modificación, nombre del archivo).

ls –R ← Busca recursivamente en subdirectorios.

ls –t ← los ordena por fecha de última modificación

- Se pueden usar combinaciones de los mismos, por ejemplo ls –al, ls –alF,

Línea de comandos (cont.)



Comando mkdir: Crear directorios

mkdir nombre_directorio

Ejemplo:

```
bash-2.05$ mkdir public_html
```

Comando pwd: mostrar ruta del directorio de trabajo actual

pwd

```
bash-2.05$ pwd
/export/home/usuarios/
```

Comando cd: cambiar de directorio de trabajo

cd .. ← Un nivel inferior

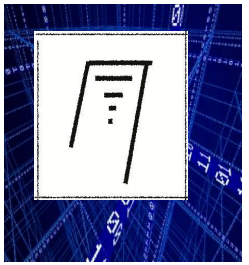
cd ~ ← Cambiarse al directorio personal

cd ~/public_html/ ← Se cambia a public_html que está en el directorio personal

cd public_html ← Se cambia al directorio public_html que está un nivel arriba de donde se está actualmente

cd /usr/bin ← Se mueve a la raíz de ahí ingresa al directorio usr y de ahí al directorio bin

Línea de comandos (cont.)

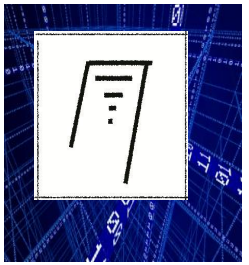


- **cd ../.. ← ¿?**
- **cd ← ¿?**

- **Comando chmod:** Cambiar permisos

- chmod UGO nombre_archivo
- U, G, O corresponden a los permisos para el usuario actual (User) el grupo actual (Group) y otros usuarios (Other).
- El número que se da corresponde a una combinación en binario de los permisos que se les desea asignar en el siguiente orden:
  - RWX (Read, Write, Execute)
- Por ejemplo si el usuario desea tener todos los permisos, deberá asignarse R=1, W=1 y X=1 (111_2=7), si para el grupo al que está asignado se le quiere dar permisos de R y W pero a los demás sólo de lectura deberá ser G=101_2=5 y O=100_2=4. → 754
- `bash-2.05$` chmod 754 public_html

Línea de comandos (cont.)



- **Comando find:** Búsqueda de archivos.

bash-2.05$ **find /usr/bin -name ‘perl’ -print**

/usr/bin/perl

➢ Buscar en el directorio /usr/bin/ el comando de nombre perl y mostrarlo en pantalla

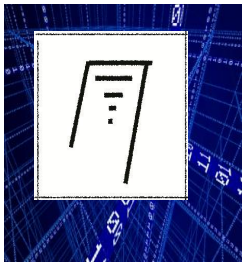
bash-2.05$ **find /usr/bin -name 'pro*' -print**

/usr/bin/profiles

/usr/bin/projects

/usr/bin/prodreg

Línea de comandos (cont.)



- **Cat ← Muestra el contenido de un archivo de texto**
  - n para mostrar el número de línea

bash-2.05$ **cat -n HelloWorld.java**

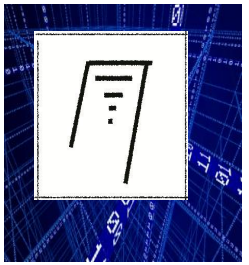
```
1  public class HelloWorld {
2
3          public static void main(String[] args) {
4                  java.lang.System.out.println("Hello World!");
5          }
6
7  }
```

Es posible usar el comando cat para mostrar el contenido de más de un archivo al mismo tiempo:

cat hola.txt texto2.txt texto3.txt *.c

- **Se puede ejecutar el comando cat, pero enviar la salida a un archivo:**
  - **cat HelloWorld.java > ejemplo.txt ← Genera un nuevo archivo**
  - **cat HelloWorld.java >> ejemplo.txt ← Agrega al final del archivo especificado**

Línea de comandos (cont.)

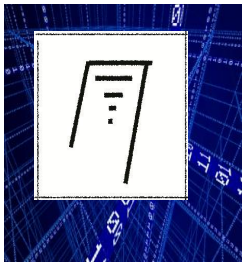


- **Comando more:** igual que el anterior, muestra el archivo, pero por páginas.
  - Con enter avanza línea a línea, espacio para avanzar página por página
  - Ejemplo:
    - **more archivolargo.txt**

- **Comando less:** igual que more, pero permite usar el cursor (flechas) para navegar en el archivo.
  - Movimiento ← flechas
  - Para salir escribir ← **:q**
  - M ← muestra en que línea se encuentra
  - Ejemplo
    - **less archivolargo.txt**

- **Comando wc:** Cuenta cuantas líneas, palabras y caracteres hay en un archivo.

Línea de comandos (cont.)



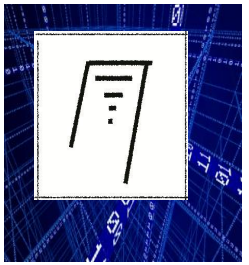
- **Es posible direccionar la salida de cualquier comando a otros comandos, por ejemplo:**

  - `ls -al | more`
  - `ls -al | less`
  - `cat archivo.txt | less`

- También es posible mostrar el

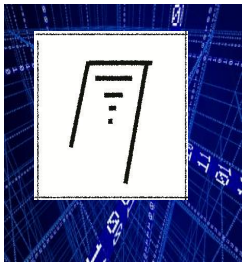
Pico – Editor de textos

- G Get Help
- ^O WriteOut
- ^R Read File
- ^Y Prev Pg
- ^K Cut Text
- ^C Cur Pos
- ^X Exit
- ^J Justify
- ^W Where is
- ^V Next Pg
- ^U UnCut Text
- ^T To Spell



MANAGING FILES FROM THE COMMAND LINE

Managing files



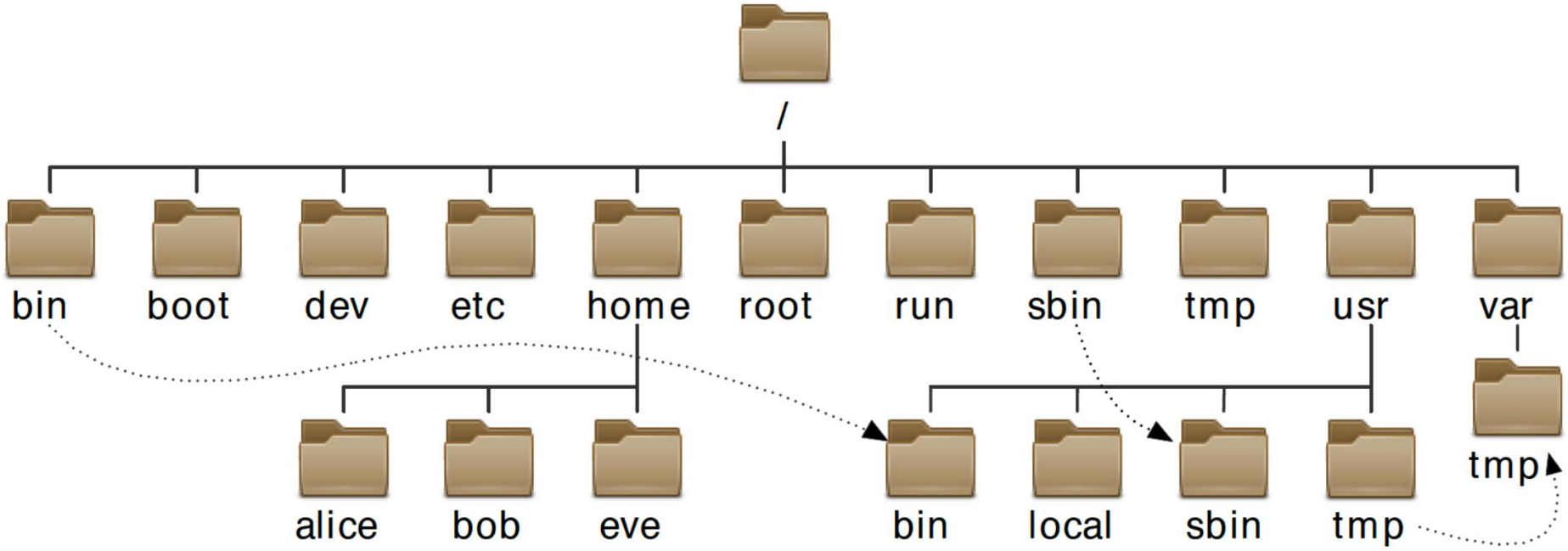
Todos los sistemas Linux/UNIX organizan su sistema de ficheros en forma de árbol invertido de directorios, llamado “jerarquía del sistema de ficheros”.

Dicho árbol, oculta la complejidad del almacenamiento empleado para crearlo.

Todo árbol tiene un comienzo a partir del cual se desarrolla hacia abajo. EL directorio origen del árbol se denomina “directorio raíz o root” y se representa por el símbolo “/”.

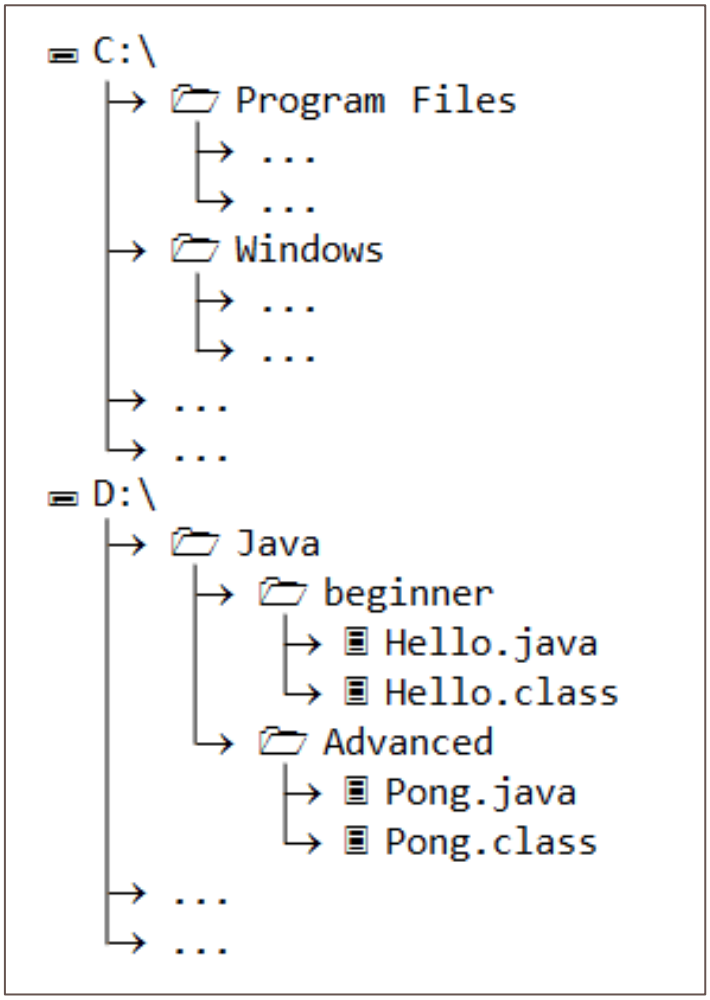
Managing files

Jerarquía del Sistema
de Ficheros LINUX

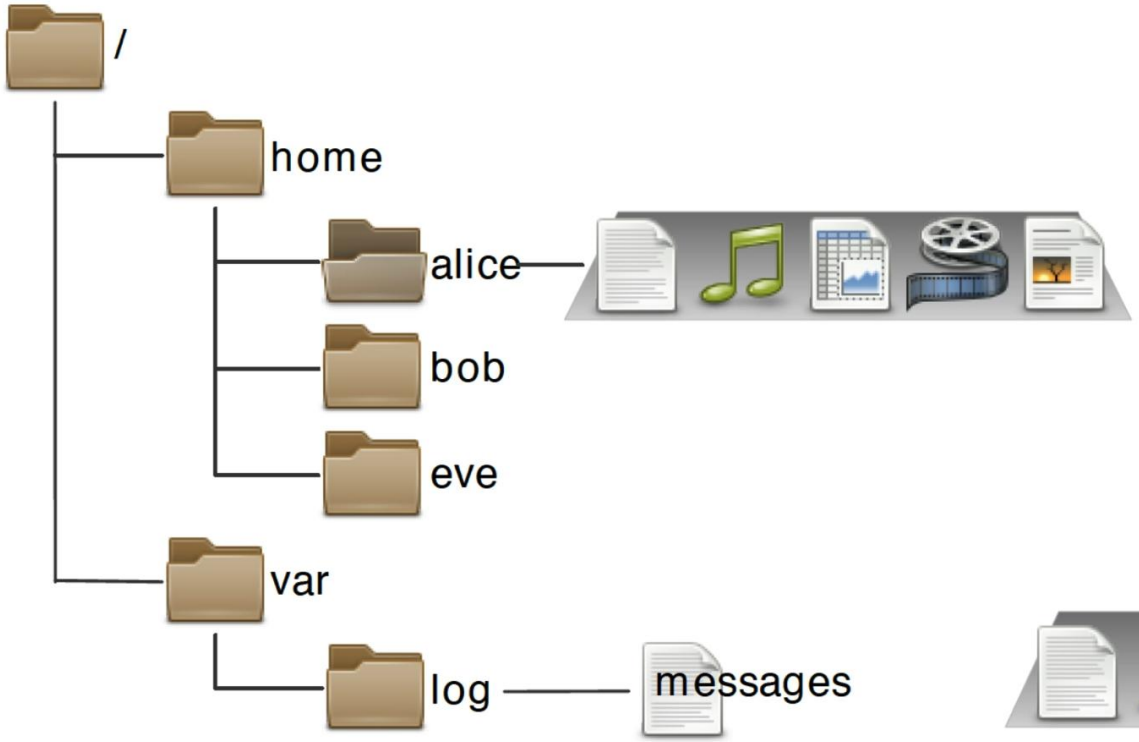


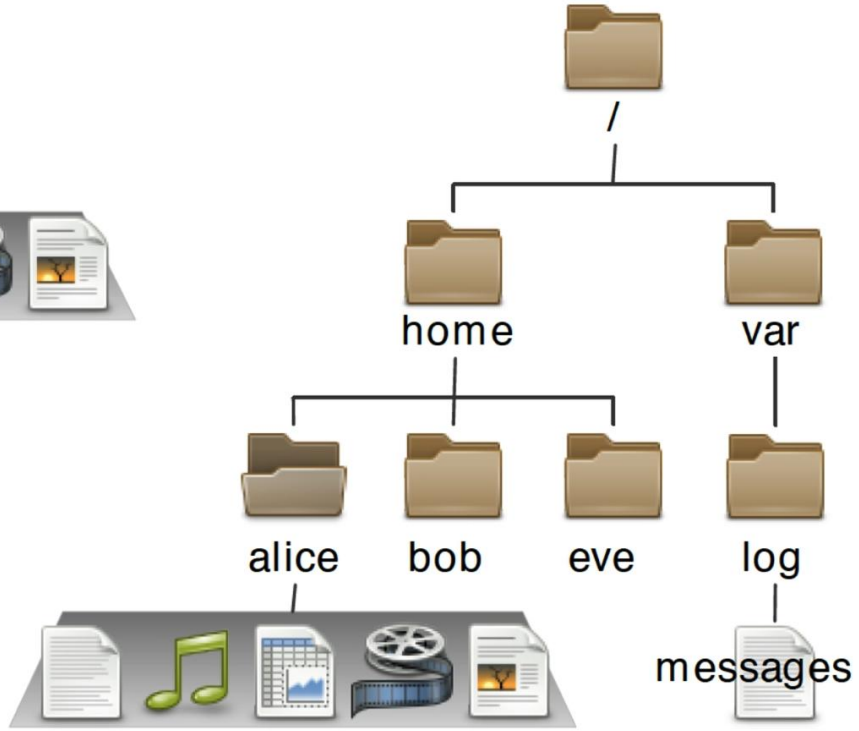
Managing files

Jerarquía del Sistema de Ficheros Windows

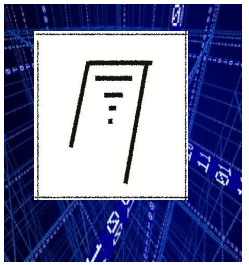


Managing files





Managing files



Directorios en el S.O UBUNTU

/bin is a place for most commonly used terminal commands, like `ls`, `mount`, `rm`, etc.

/boot contains files needed to start up the system, including the Linux kernel, a RAM disk image and bootloader configuration files.

/dev contains all device files, which are not regular files but instead refer to various hardware devices on the system, including hard drives.

/etc contains system-global configuration files, which affect the system's behavior for all users.

/home home sweet home, this is the place for users' home directories.

/lib contains very important dynamic libraries and kernel modules

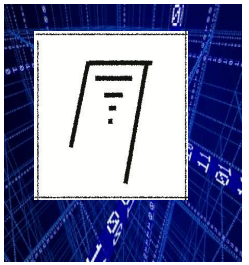
/media is intended as a mount point for external devices, such as hard drives or removable media (floppies, CDs, DVDs).

/mnt is also a place for mount points, but dedicated specifically to "temporarily mounted" devices, such as network filesystems.

/opt can be used to store additional software for your system, which is not handled by the package manager.

/proc is a virtual filesystem that provides a mechanism for kernel to send information to processes.

Managing files



Directorios en el S.O UBUNTU

/root is the superuser's home directory, not in `/home/` to allow for booting the system even if `/home/` is not available.

/run is a tmpfs (temporary file system) available early in the boot process where ephemeral run-time data is stored. Files under this directory are removed or truncated at the beginning of the boot process.
(It deprecates various legacy locations such as `/var/run`, `/var/lock`, `/lib/init/rw` in otherwise non-ephemeral directory trees as well as `/dev/.*` and `/dev/shm` which are not device files.)

/sbin contains important administrative commands that should generally only be employed by the superuser.

/srv can contain data directories of services such as HTTP (`/srv/www/`) or FTP.

/sys is a virtual filesystem that can be accessed to set or obtain information about the kernel's view of the system.

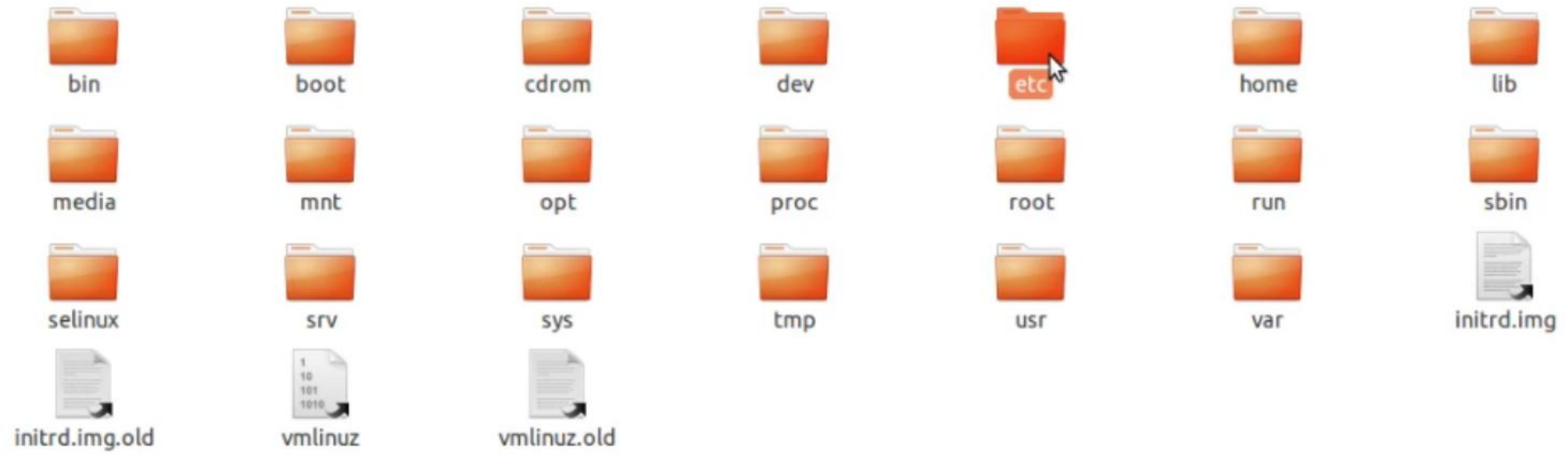
/tmp is a place for temporary files used by applications.

/usr contains the majority of user utilities and applications, and partly replicates the root directory structure, containing for instance, among others, `/usr/bin/` and `/usr/lib`.

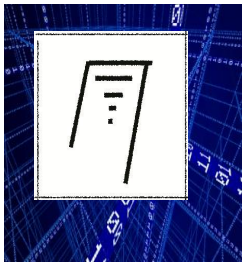
/var is dedicated to variable data, such as logs, databases, websites, and temporary spool (e-mail etc.) files that persist from one boot to the next. A notable directory it contains is `/var/log` where system log files are kept.

Managing files

Directorios en el S.O UBUNTU



Managing files

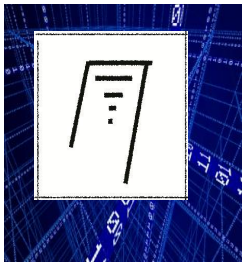


¿ Dónde estamos ?

Para averiguar en qué punto de la jerarquía de file system nos encontramos, utilizamos el comando “pwd” (**P**resent **W**orking **D**irectory) Nos devuelve la ruta donde estamos situados. Su “path absoluto”.

alumno@lab:~$ pwd

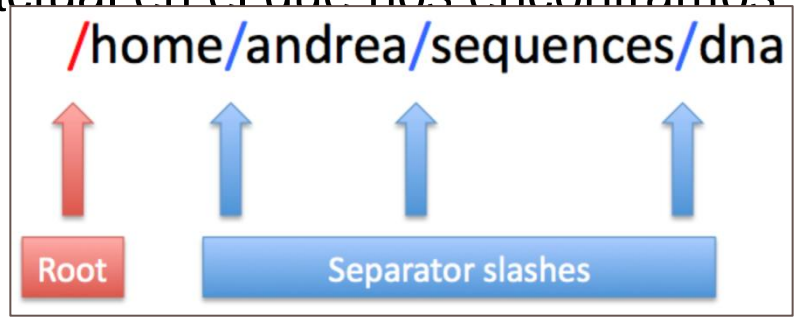
```
alumno@LAB:/usr/lib/apt/methods$ pwd
/usr/lib/apt/methods
```



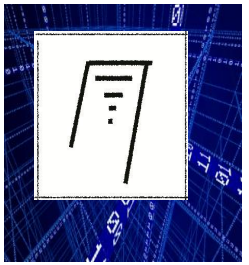
Path Absoluto vs. Path Relativo

Un PATH ABSOLUTO es una ruta totalmente cualificada desde el directorio root (***/***), especificando todos los directorios por los que se pasa, hasta donde está el directorio o el fichero al cual queremos referirnos.

Un PATH RELATIVO es la ruta que identifica a un directorio o fichero, desde el directorio actual en el que nos encontramos.



Managing files



Navegando por los paths

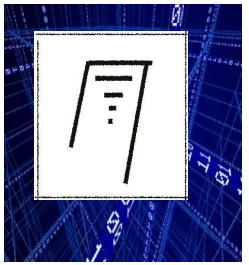
Empleamos el comando “cd” para movernos por los directorios y el comando “ls” para que el sistema nos muestre los contenidos de dichos directorios.

Generalmente, cuando nos conectamos al sistema, este nos sitúa en nuestro directorio “home”: /home/username

```
alumno@lab:~$  pwd
alumno@lab:~$  ls
alumno@lab:~$  cd /
alumno@lab:  ls
alumno@lab:  cd /var
alumno@lab:  ls
```

```
alumno@lab: cd
alumno@lab:~$  pwd
alumno@lab:~$  cd ..
alumno@lab: ls
alumno@lab: cd
alumno@lab:~$  cd ../..
```

Managing files



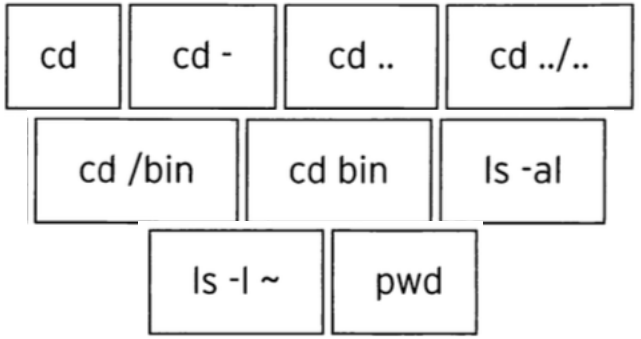
Navegando por los paths

“ls –l” nos muestra información extendida del contenido de un directorio.
“ls –a” nos muestra todos los ficheros y directorios, incluido los ocultos.
“ls –R” nos muestra el contenido de un directorio de forma recursiva, incluyendo el contenido de los subdirectorios.
“cd –“ nos sitúa en el directorio, de nuestra ubicación anterior y nos permite fácilmente conmutar entre dos directorios de trabajo

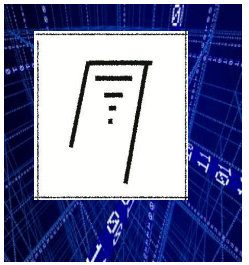
alumno@lab:~$ cd ; pwd
alumno@lab:~$ cd ..
alumno@lab: cd /usr ; ls –la
alumno@lab: cd –
alumno@lab: ls –la
alumno@lab: cd

Managing files

Action to accomplish	Command
List the current user's home directory (long format) in simplest syntax, when it is not the current location.	
Return to the current user's home directory.	
Determine the absolute path name of the current location.	
Return to the most previous working directory.	
Move up two levels from the current location.	
List the current location (long format) with hidden files.	
Move to the binaries location, from any current location.	
Move up to the parent of the current location.	
Move to the binaries location, from the root directory.	



Managing files



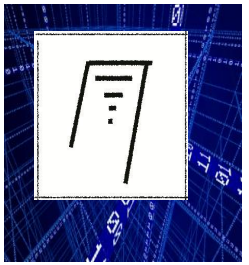
Crear Directorios y Ficheros

“mkdir directorio” nos permite crear un directorio.
“mkdir –p dir1/dir2” nos permite crear un árbol de directorios.
“touch fichero.extensión” nos permite crear un fichero

El target de “mkdir” y “touch” puede ir acompañado de un path absoluto o relativo.

Podemos crear múltiple ficheros y directorios en una única sentencia

“mkdir dir1 dir2 dir3 …. “
“touch file1.txt file2.txt file3.txt … “



Crear Directorios y Ficheros

- Ejercicio:

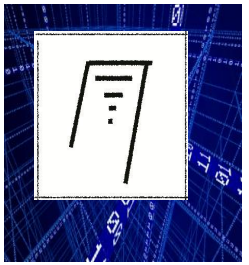
Crear los siguientes directorios en nuestro home/user:

documentos
musica
imagenes
videos

Crear los siguiente ficheros:

en /home/user/documentos : file1.txt y file2.txt
en /home/user/imagenes: img1.jpg e img2.jpg

Managing files



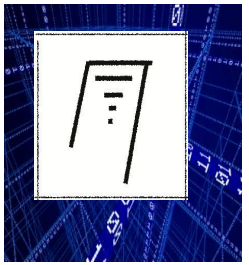
Administrar Directorios y Ficheros

Con el comando “cp file1 file2” copiamos un fichero.
Con el comando “cp –r dir1 dir2” copiamos un directorio.

Con el comando “mv file1 file2” movemos o renombramos un fichero.
Con el comando “mv dir1 dir2” movemos o renombramos un directorio.

Con el comando “rm file1” eliminamos un fichero.
Con el comando “rm –r dir1” eliminamos un directorio.
El comando “rm –rf dir” fuerza el borrado del directorio.
El comando “rm –ri dir” pide confirmación para el borrado del directorio.

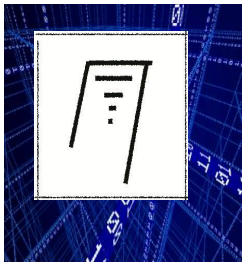
Managing files



Wildcars

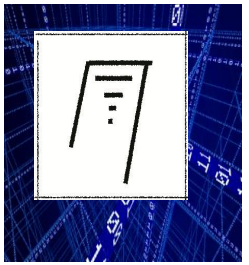
Nos sirve para manipular múltiples ficheros al mismo tiempo.
Principalmente se utilizan: * , ? , [abc] [1–10] , !

“*” Sustituye a cualquier número o carácter.
“?” Sustituye a un único carácter.
“[*characters*]” coincidencia con cualquiera de los caracteres mostrados.
“[num1-num2]” coincidencia con un rango de valores.
“[! *characters*]” coincidencia con ninguno de los caracteres mostrados.



GETTING HELP

GETTING HELP



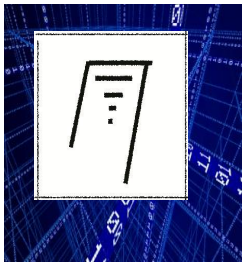
MAN

Es una herramienta de sistemas Linux/Unix que se utiliza para documentar y aprender sobre comandos, archivos, llamadas de sistema, etc…

El manual está dividido en diferentes secciones y un mismo término puede encontrarse en diferentes de ellas.

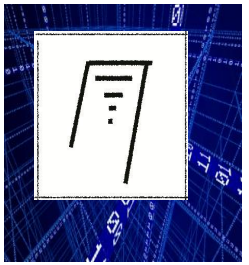
Por ejemplo: passwd
Puede referirse al comando que permite cambiar el password o bien al fichero que almacena las cuentas de usuario local /etc/passwd.

GETTING HELP



Sección	Descripción
1	Comandos Generales
2	Llamadas al sistema
3	Biblioteca C de funciones
4	Ficheros especiales (normalmente dispositivos, que se pueden encontrar en /dev) y drivers
5	Formatos de fichero y convenciones
6	Juegos y salvapantallas
7	Miscelánea
8	Comandos de administración del sistema y Demonios

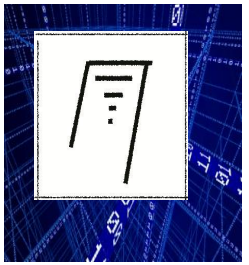
GETTING HELP



alumno@lab:~$ man passwd
alumno@lab:~$ man –k passwd
alumno@lab:~$ whatis passwd alumno@lab:~$ man –f passwd
alumno@lab:~$ man 1 passwd
alumno@lab:~$ man 5 passwd

```
alumno@LAB:~$ man -k passwd
update-passwd (8)    - Actualiza /etc/passwd, /etc/shadow y /etc/group de forma segura
chgpasswd (8)        - update group passwords in batch mode
chpasswd (8)         - update passwords in batch mode
gpasswd (1)          - administer /etc/group and /etc/gshadow
grub-mkpasswd-pbkdf2 (1) - generate hashed password for GRUB
pam_localuser (8)    - require users to be listed in /etc/passwd
passwd (1)           - change user password
passwd (1ssl)        - compute password hashes
passwd (5)           - the password file
alumno@LAB:~$
```

GETTING HELP

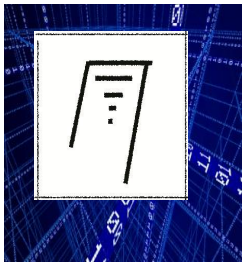


```
alumno@LAB:~$ whatis passwd
passwd (1)           - change user password
passwd (1ssl)        - compute password hashes
passwd (5)           - the password file
alumno@LAB:~$
```

```
PASSWD(1)                                        User Commands

NAME
       passwd - change user password

SYNOPSIS
       passwd [options] [LOGIN]
```

```
PASSWD(5)                              File Formats and Conversions

NAME
       passwd - the password file
```

GETTING HELP



Documentación en /usr/share/docs

Además de incluir documentación en “man” y en “info”, se suele incluir documentación en /usr/share/docs.

Generalmente, cuando se instala un paquete, este crea un directorio con su nombre en dicha ruta y almacena en dicho directorio su documentación.

alumno@lab:~$ cat /usr/share/docs/adduser/examples/adduser.local

alumno@lab:~$ cat /usr/share/docs/adduser/examples/adduser.local |more
alumno@lab:~$ cat /usr/share/docs/adduser/examples/adduser.local |less

GETTING HELP

Ayuda on-line

https://www.ubuntu.com/support/community-support

Community support

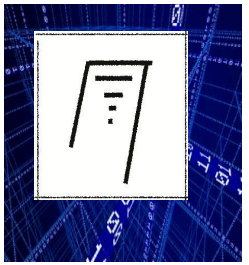
You can find support from a variety of sources. Take a look – you're likely to find an answer to every question. If you can't find an answer, just ask the people in our active forums.

Get free community support

Read the docs

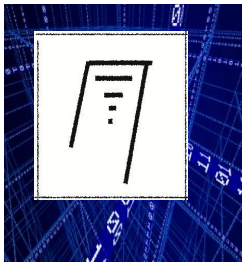
Technical answers system

Ubuntu security notices ›



CREATING, VIEWING & EDITING TEXT FILES

CHAPER 4



Entrada Estándar y Salidas Estándar y Estándar Error

Bajo condiciones normales cualquier programa Unix tiene tres flujos de E/S abiertos cuando arranca: uno para la entrada, otro para la salida y otro para imprimir los mensajes de error. Channels 0, 1 y 2.

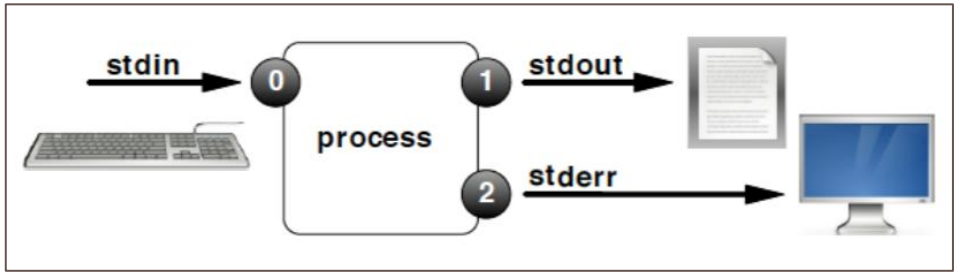
Estos flujos están ligados al terminal de usuario (tty).

El flujo de entrada se conoce como "entrada estándar"; el flujo de salida como "salida estándar"; y el flujo de error como "error estándar".

Estos términos se abrevian para formar los símbolos utilizados para referirse a esos ficheros, esto es, stdin, stdout y stderr.

CREATING, VIEWING & EDITING TEXT FILES

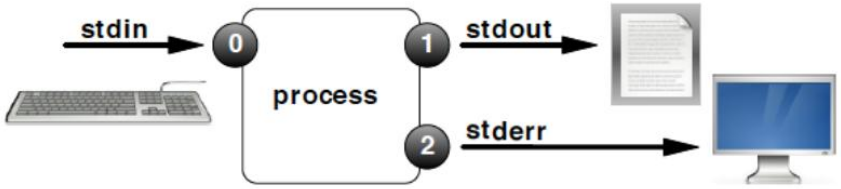
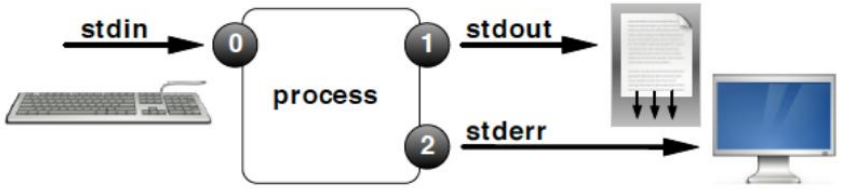
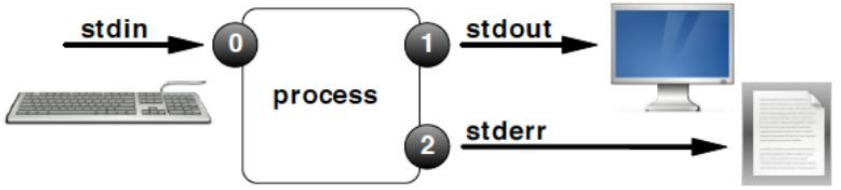
Entrada Estándar y Salidas Estándar y Estándar Error



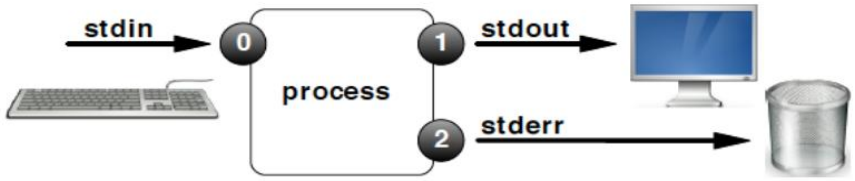
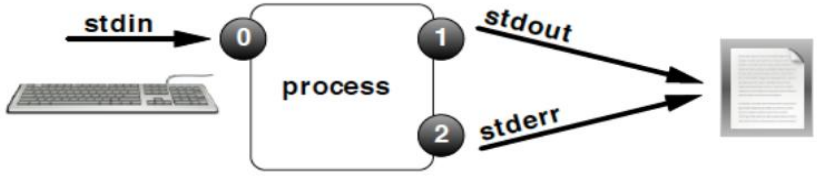
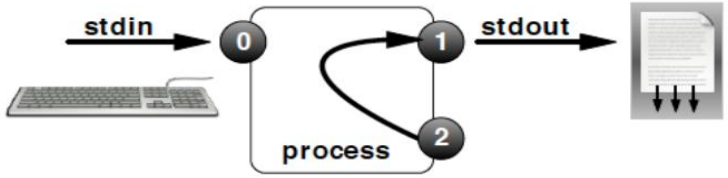
Channels (File Descriptors)

Number	Channel name	Description	Default connection	Usage
0	**stdin**	Standard input	Keyboard	read only
1	**stdout**	Standard output	Terminal	write only
2	**stderr**	Standard error	Terminal	write only

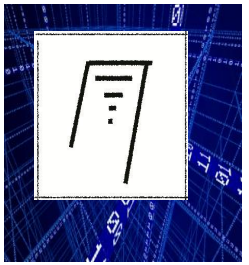
CREATING, VIEWING & EDITING TEXT FILES

Usage	Explanation (note)	Visual aid
`>file`	redirect **stdout** to a file (1)	
`>>file`	redirect **stdout** to a file, append to current file content (2)	
`2>file`	redirect **stderr** to a file (1)	

CREATING, VIEWING & EDITING TEXT FILES

Usage	Explanation (note)	Visual aid
2>/dev/null	discard **stderr** error messages by redirecting to **/dev/null**	
&>*file*	combine **stdout** and **stderr** to one file (1)	
>>*file* 2>&1	combine **stdout** and **stderr**, append to current file content (2) (3)	

CREATING, VIEWING & EDITING TEXT FILES

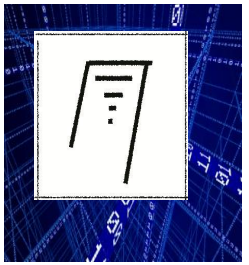


Redirección bi-direccional

alumno@lab:~$ ls –l > file10.txt

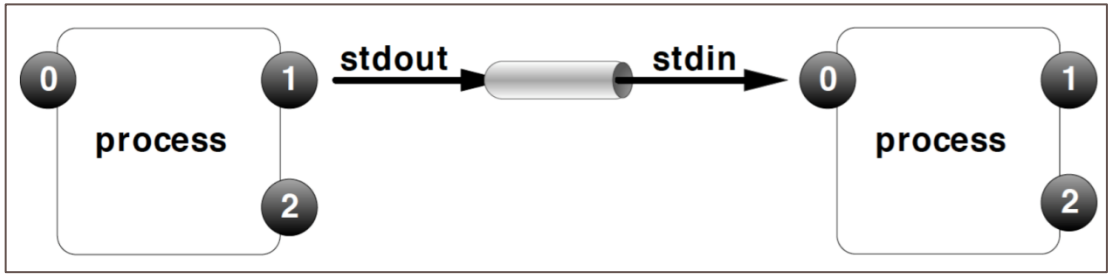
alumno@lab:~$ wc –l < file10.txt

alumno@lab:~$ wc –l < file10.txt > file-count.txt

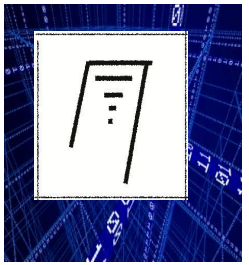


Construcción de Pipes

Un “Pipe” envía la salida de un channel a otro proceso. Se emplea el símbolo “|”
A diferencia de la redirección, que reenvía hacia o desde un fichero, una determinada salida.



CREATING, VIEWING & EDITING TEXT FILES



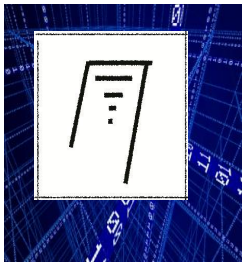
Construcción de Pipes

Un “Pipe” envía la salida de un channel a otro proceso. Se emplea el

alumno@lab:~$ ls –l /usr/bin | less

alumno@lab:~$ ls | wc –l > how-many-files.txt

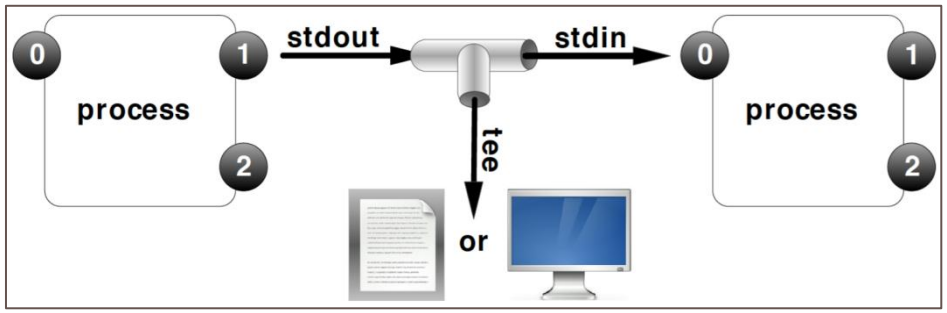
alumno@lab:~$ ls –t | head –n 10 > ten-last-changed-files.txt



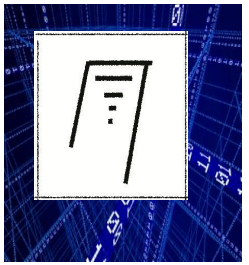
Construcción de Pipes con Tee

Un “Pipe Tee” envía la salida de un channel a otro proceso, pero permite mostrar el resultado dentro de la redirección.

Command 1 | tee Command 2 | Command 3



CREATING, VIEWING & EDITING TEXT FILES



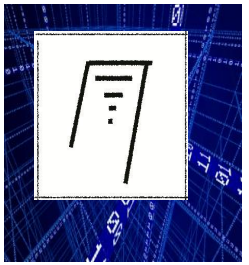
Construcción de Pipes con Tee

alumno@lab:~$ ls –l

alumno@lab:~$ tty
/dev/tty1

alumno@lab:~$ ls –l | wc –l > salida1.txt

alumno@lab:~$ ls –l | tee /dev/tty1 | wc –l > salida1.txt



Result needed	Redirection syntax used
Display command output to terminal, ignore all errors.	
Send command output to file; errors to different file.	
Send output and errors to the same new, empty file.	
Send output and errors to the same file, but preserve existing file content.	
Run a command, but throw away all possible terminal displays.	
Send command output to both the screen and a file at the same time.	
Run command, save output in a file, discard error messages.	

&>/dev/null

&>file

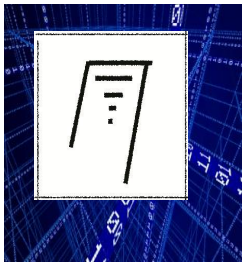
2>/dev/null

> file 2> /dev/null

>>file 2>&1

>file 2>file2

| tee file



Construcción de Múltiples Comandos

1. Si queremos ejecutar dos comandos consecutivos:

command1 ; command2

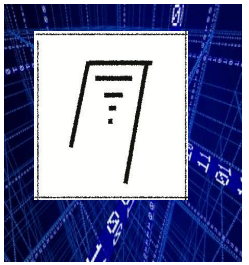
2. Si queremos ejecutar el command1 correctamente, antes de ejecutar el command 2:

command1 && command2

3. Si queremos que el command2 se ejecute si el command1 falla:

command1 || command2

CREATING, VIEWING & EDITING TEXT FILES



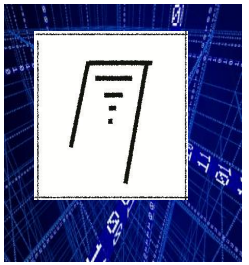
Construcción de Múltiples Comandos

alumno@lab:~$ ls –la ; cd .. ; pwd

alumno@lab:~$ cd /usr/invento && ls –la

alumno@lab:~$ cd /usr/invento || ls –la

EDITOR DE TEXTO “VI”



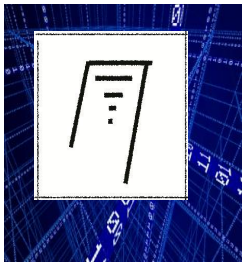
Vi (*Visual*) es un programa que entra en la categoría de los editores de texto. A diferencia de un procesador de texto, no ofrece herramientas para determinar visualmente cómo quedará el documento impreso.

Carece de opciones como centrado o justificación de párrafos, pero permite mover, copiar, eliminar o insertar caracteres con mucha versatilidad.

El editor vi tiene dos modos de operación:

Modo de comandos
Modo insertar

EDITOR DE TEXTO “VI”



Modo de comandos:

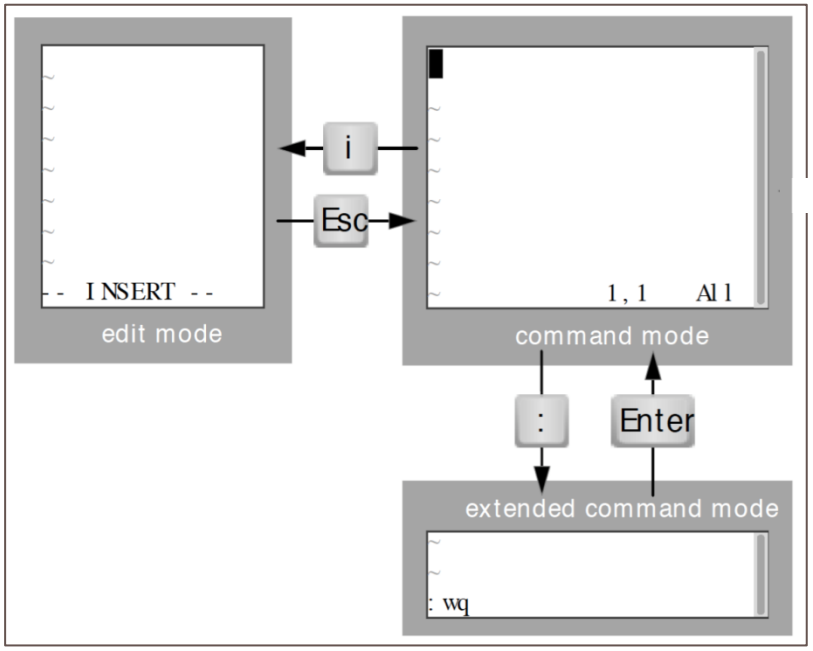
En el modo de comandos, podemos desplazarnos dentro de un archivo y efectuar operaciones de edición como buscar texto, eliminar texto, modificar texto, etc. Vi suele iniciarse en modo de comandos.

Modo insertar:

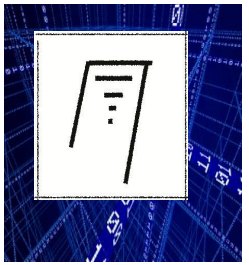
En el modo insertar, podemos escribir texto nuevo en el punto de inserción de un archivo.

Para volver al modo de comandos, presione la tecla “esc”.

EDITOR DE TEXTO “VI”



EDITOR DE TEXTO “VI”



Editar y modificar

- Para insertar texto antes del cursor: `i`
- Para insertar texto después del cursor: `a`
- Para insertar texto al principio de la línea: `I`
- Para insertar texto al final de la línea: `A`

Copiar y pegar

- Para copiar la línea actual: `y` `y`
- Para copiar una palabra: `y` `w`
- Para copiar 7 líneas: `y` `7` `y`
- Para pegar después del cursor: `p`
- Para pegar antes del cursor: `P`

Borrar

- Para borrar un caracter: `x` ó `X`
- Para borrar la línea actual: `dd`

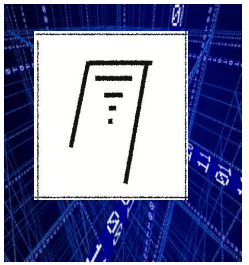
Abrir, guardar y salir

- Para abrir un archivo: `:` `e` `nombre_de_archivo`
- Para guardar los cambios y salir: `:` `x`
- Para salir: `:` `q`
- Para salir sin guardar (forzar la salida): `:` `q` `!`

Una de las utilidades más comunes es el uso de `:wq`

`:q!` que sale de vi sin guardar cambios.

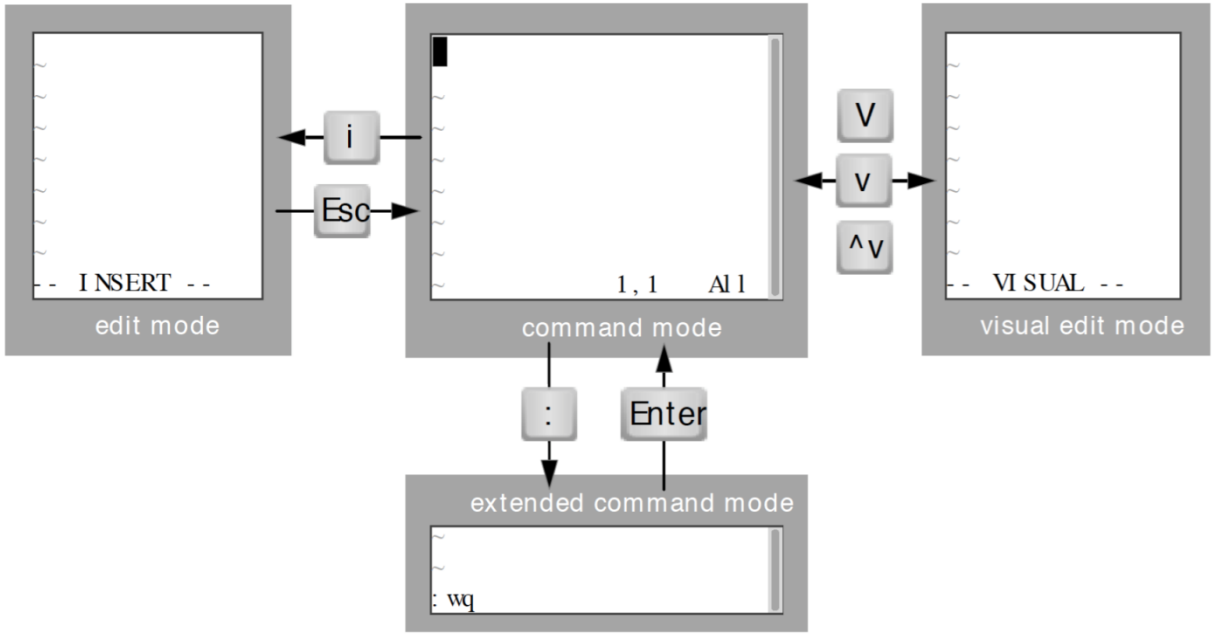
EDITOR DE TEXTO “VIM”



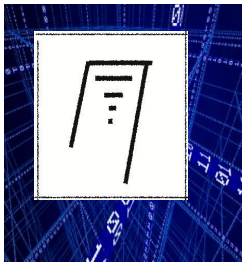
Vim (*Vi IMproved*) es una versión mejorada del editor de texto vi, presente en todos los sistemas UNIX.

Modo línea de Órdenes:
A este modo se accede pulsando la tecla dos puntos **:**.
Modo Visual:
Este modo es una mejora respecto a vi
Modo Selección:
Este modo empieza como el modo visual y tras la selección de un bloque de texto, se puede cambiar al modo selección mediante **Control-G.** Finaliza pulsando la tecla Escape.
Modo Ex:
Este modo se asemeja al modo línea de órdenes, con la diferencia de que tras la ejecución de una orden no se vuelve al modo comandos. Se entra en este modo pulsando **Q** y se termina con **vi**.

EDITOR DE TEXTO “VIM”



EDITOR DE TEXTO “NANO”



“nano” (oficialmente GNU nano) es un editor de texto para sistemas Unix.

Está orientado a un manejo desde teclado específicamente a combinaciones de la tecla Control.

Por ejemplo, Control-O guarda el archivo actual y Control-W abre el menú de búsqueda.

La barra de accesos directos de “nano” tiene dos filas colocadas en la parte baja de la pantalla, que lista algunos de los comandos disponibles en función del contexto.

Para ver una lista completa basta con presionar Control-G y se obtiene una pantalla de ayuda.