

Documentos Aula SUN UCM

*Aula
Sun*



UCM



CURSO AVANZADO LINUX

26 Mayo 2008

Autores:

Sergio Velasco

Alicia Martín

Aula SUN UCM de Software Libre

ÍNDICE

1-	<u>Terminal y modo gráfico.....</u>	3
2-	<u>Gestor de arranque.....</u>	7
3-	<u>Creación de usuarios.....</u>	8
4-	<u>Configuración de la red.....</u>	8
5-	<u>Editar archivos de texto.....</u>	11
	5.1- <u>Editor Vim.....</u>	11
6-	<u>Instalar programas.....</u>	13
	6.1- <u>Instalación desde repositorios.....</u>	14
	6.2- <u>Descarga y compilación.....</u>	14
7-	<u>Permisos y propietarios.....</u>	15
8-	<u>Comandos de la máquina.....</u>	17
9-	<u>Copiar archivos.....</u>	18
10-	<u>Escritorio remoto.Vnc.....</u>	19
11-	<u>Conectarse a una máquina de Linux desde Windows.....</u>	21
12-	<u>Servidor de paginas web.....</u>	21
13-	<u>Scripts.....</u>	22
14-	<u>Demonios.....</u>	23
15-	<u>Firewall e Iptables.....</u>	25
	15.1- <u>Firewall para nuestra máquina.....</u>	27
	15.2- <u>Firewall para una red local con salida a internet.....</u>	28
	15.3- <u>Otro ejemplo de Firewall.....</u>	30
	15.4- <u>Logueo en red sin contraseña.....</u>	30
16-	<u>Copias de seguridad.....</u>	31
	16.1- <u>Backups de directorios y ficheros.....</u>	31
	16.2- <u>Backup de recuperación.....</u>	33
	16.3- <u>Recomendaciones.....</u>	34
17-	<u>Anexo:Comandos de interés.....</u>	34

CURSO AVANZADO LINUX

AULA SUN UCM

Este curso está orientado a personas que ya hayan tenido algún contacto con Linux, pero que deseen alcanzar un mayor conocimiento sobre su funcionamiento. Se trata de un curso de un nivel básico, pero en profundidad, donde trataremos la administración de usuarios y de archivos, así como algunos comandos que serán de gran interés.

1- Terminal y modo gráfico

Cuando iniciamos una sesión en las máquinas, lo hace por defecto en modo gráfico. Desde aquí, ya se ha visto en un curso introductorio como abrir una terminal, pero existe otro modo de trabajar en terminal sin necesidad de acceder primero al entorno gráfico.

Existen 7 terminales, a cada una de las cuales accederemos con la combinación de teclas *Alt Ctrl F1* (*F2, F3.....*). Solo la *F7* es la terminal gráfica, el resto serán terminales de texto.

Si deseamos salir de una sesión en terminal de texto, solo tendremos que teclear el comando *exit*.

Shell

El *shell* es el interprete de comandos. En DOS normalmente el *shell* es el *command.com*, en UNIX existen muchos *shell* usados habitualmente. Desde la séptima edición de UNIX el *shell* por excelencia es el *sh*. Fue escrito por Steven Bourne, y es por eso que se lo suele llamar Bourne Shell. Está disponible en todas las versiones de UNIX y es lo suficientemente básico como para que funcione en todas las plataformas.

csh Un *shell* un poco mejor con respecto al *sh* es el *csh*, que fue escrito por Bill Joy, y debe su nombre, al lenguaje de programación C. Al hacer scripts en este *shell* puede utilizarse una sintaxis similar a la de C.

ksh Otro *shell*, que como ventaja maneja un historial de comandos, es el *ksh* (korn **shell**). Está basado en *sh*, con algunos agregados muy básicos para hacerlo más amigable.

bash Uno de los *shell* más avanzados, muy popular en la comunidad GNU/Linux. El nombre significa *Bourne Again Shell*. Tiene licencia GNU y se suele incluir como *shell* predeterminado en las distribuciones. Ofrece las mismas capacidades que *csh*, pero incluye funciones avanzadas, tanto para el usuario como para el programador. En particular, podremos acceder a un historial de los comandos ejecutados, que se conserva incluso al pasar de una sesión a otra, utilizando los cursores.

Hay muchas otras versiones de *shell* además de estas. Pero el estilo general de todos es muy similar.

TERMINAL	PROMPT
bash	<i>usuario@sunny01:~\$</i>
sh	<i>\$</i>
tcsh	<i>sunny01:~\$</i>
csh	<i>%</i>

El *prompt* nos ayuda a saber en que tipo de terminal estamos.

Terminal *bash*:

~ Indica que estamos en el home de nuestro usuario

pwd Podemos ver en que directorio nos encontramos

Es útil recordar comandos como:

ls lista los archivos en un directorio.

cd para entrar en una carpeta.

cd .. para salir de un directorio.

Tabulador Nos completa un comando.

clear Limpia la terminal.

cp Copia archivos a un nuevo emplazamiento.

w Sirve para ver que usuarios estan conectados.

Hay que tener precaución, porque Linux distingue entre mayúsculas y minúsculas, así que no será lo mismo el usuario de nombre *Usuario* que el de nombre *usuario*.

Otros comandos quizá menos utilizados pero también útiles son:

mv Muy similar a *cp*, el comando *mv* es el que se utiliza para mover archivos de un lugar a otro, o para cambiarle el nombre a un archivo. Si ejecutamos, *mv viejo nuevo*, el archivo *viejo* habrá pasado a llamarse *nuevo*.

Por otro lado, si ejecutamos *mv archivo1 archivo2 directorio*, los archivos *archivo1* y *archivo2* se moverán dentro de *directorio*.

du El comando *du*, *Disk Usage*, nos muestra el espacio que ocupan todos los directorios a partir del directorio actual. El número de la primera columna es el espacio ocupado por el directorio y está expresado en *kb*.

du -s nos muestra únicamente el total.

du -a muestra lo que ocupan los archivos, además de los directorios.

du -h hace el listado, indicando la unidad (*human readable*).

du archivo nos dice cuánto ocupa el archivo.

```
npebm@sunny08: ~
Archivo Editar Ver Terminal Solapas Ayuda
npebm@sunny08:~$ du SoftLibreCiencia.pdf
560    SoftLibreCiencia.pdf
npebm@sunny08:~$ du sun.odt
0      sun.odt
npebm@sunny08:~$ du -s
1703828 .
npebm@sunny08:~$
```

find El comando *find* permite encontrar archivos, utilizando diversas técnicas. En principio, si se le pasa como parámetro únicamente una determinada ruta, por ejemplo *find /home/user*, el comando buscará todos los archivos y directorios que se encuentren a partir de esa ruta. Utilizando algunos otros parámetros es posible buscar los archivos por diversos criterios.

find . -name "hola.txt" encuentra todos los archivos llamados hola.txt que se encuentren a partir del directorio actual. Las comillas no son obligatorias, pero son recomendables si se quieren usar opciones más complejas (por ejemplo, utilizando metacaracteres de *shell*, que se explican en la sección 4.1).

find -size 50k busca los archivos que ocupan 50 kilobytes a partir directorio actual.

find -size 20c, buscará los archivos que ocupen 20 bytes.

find -size 5b, buscará los archivos que ocupen 5 bloques de 512 bytes cada uno.

find /home/user -empty busca todos los archivos que se encuentran vacíos a partir del directorio */home/user*.

```
npebm@sunny08: ~
Archivo Editar Ver Terminal Solapas Ayuda
npebm@sunny08:~$ find -size 50k
./Desktop/lib/freehep-util-2.0.1.jar
./Desktop/src/cytoscape/data/writers/XGMLWriter.java
./mozilla/firefox/9a0zfs3p.default/Cache/52128A08d01
./mozilla/firefox/9a0zfs3p.default/Cache/C2E1A4E4d01
npebm@sunny08:~$ find -size 20c
./mozilla-thunderbird/i6v3tfuh.default/lock
./mozilla/firefox/9a0zfs3p.default/lock
npebm@sunny08:~$
```

cat Ejecutando *cat archivo* podremos ver el contenido de archivo. Este comando puede recibir una serie de archivos, y el resultado será que nos mostrará un archivo a continuación del otro. Un caso especial se produce cuando ejecutamos *cat* sin ningún nombre de archivo. En este caso, el comando esperará a que nosotros le demos una entrada, y la irá reproduciendo línea por línea. Hasta que presionemos la combinación *Ctrl-d*, que indica que la entrada ha terminado.

od El comando *od Octal Dump*, nos permite ver byte a byte el contenido de un archivo. La primera columna es la dirección de cada línea que vemos. Utilizando las distintas opciones, podemos visualizarlo en varios formatos:

od archivo nos muestra el contenido del archivo expresado en números octales, generalmente tomados de a dos bytes.

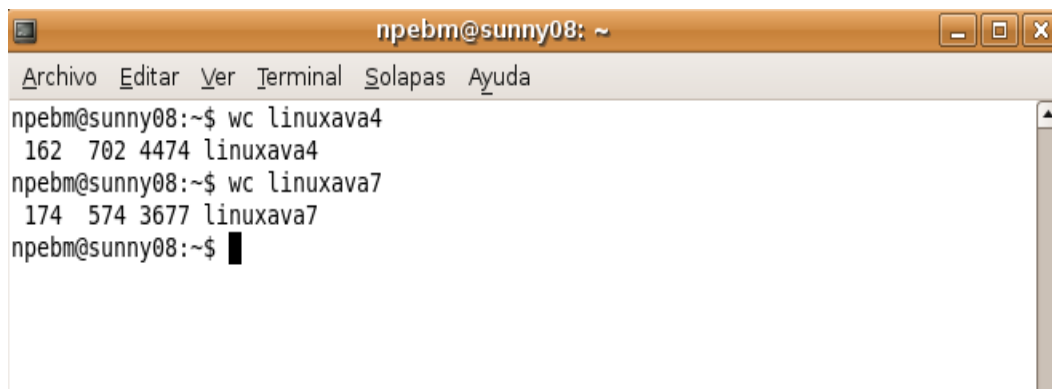
od -b archivo nos muestra el contenido, en números octales, byte a byte.

od -c archivo nos muestra los caracteres que forman el archivo, uno por uno.

od -cb archivo nos muestra los caracteres, y debajo de cada carácter el número octal del byte.

od -h archivo nos muestra el contenido, en números hexadecimales, tomados de a dos bytes.

wc El comando *wc archivo*, se utiliza para contar la cantidad de líneas, palabras y letras que tiene un archivo.



```
npebm@sunny08: ~
Archivo Editar Ver Terminal Solapas Ayuda
npebm@sunny08:~$ wc linuxava4
162  702 4474 linuxava4
npebm@sunny08:~$ wc linuxava7
174  574 3677 linuxava7
npebm@sunny08:~$
```

less El comando *less* permite paginar la salida de otros comandos, o bien, el contenido de algún archivo.

less archivo veremos la primera página del archivo. Si este archivo es lo suficientemente largo, podremos movernos hacia abajo y hacia arriba utilizando *PageUp*, *PageDown*, *Home*, *End*, *Enter*, los cursores, la barra espaciadora, etc. También podemos realizar búsquedas dentro del archivo, para ello utilizamos la barra invertida \, seguida del patrón que queremos buscar. Por ejemplo, si tecleamos *\consola*, nos mostrará la primera ocurrencia del patrón *consola*. Para ver la siguiente ocurrencia, utilizamos *n*, y para ver la ocurrencia anterior *N*. Para salir, utilizamos *q*.

dpkg Sirve para obtener detalles de programas que le indiquemos, como puede ser su ubicación, su tamaño, la versión que tenemos instalada....veamos ejemplos de uso:

dpkg -S bin/programa Nos dirá donde está instalado *programa*

dpkg -l firefox Nos dice la versión del programa *firefox* que tenemos instalada

dpkg -s firefox Nos muestra muchos detalles, como cuánto ocupa el programa o qué librerías usa

```
npebm@sunny08: ~
Archivo Editar Ver Terminal Solapas Ayuda
npebm@sunny08:~$ dpkg -S bin/firefox
firefox: /usr/bin/firefox
firefox-dev: /usr/bin/firefox-config
npebm@sunny08:~$ dpkg -l firefox
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-f/Unpacked/Failed-cfg/Half-inst/t-aWait/T-pend
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Nombre          Versión          Descripción
+++-----+-----+-----+
ii  firefox            2.0.0.13+1nobi  lightweight web browser based on Mozilla
npebm@sunny08:~$
```

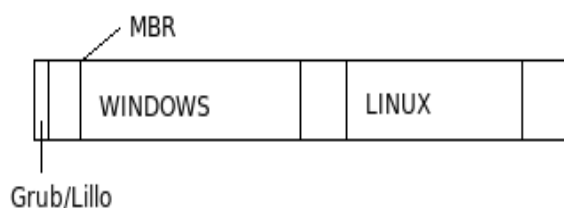
update-alternatives Permite establecer por defecto que versiones de ciertos programas queremos que utilice nuestra máquina. Sólo se puede utilizar como *root* o como usuario *sudable*.

2- Gestor de arranque:

Cuando instalemos Linux en un ordenador que tenía ya otro sistema operativo, por ejemplo Windows, se nos abrirá un gestor de arranque que nos permitirá elegir con cual de los dos sistemas queremos iniciar nuestra sesión.

Los dos gestores de arranque más habituales son **Grub** y **Lillo**

La forma de nuestro disco duro puede ser la siguiente:



Si reinstalamos Windows en nuestra máquina, perderemos el gestor de arranque. En este caso, para recuperarlo debemos reiniciar Linux con un Live-CD y desde la terminal ejecutar:

```
apt-get install grub
```

```
set up
```

Este gestor de arranque se albergará en `cd /boot/grub`

```
npebm@sunny06: /boot/grub
Archivo Editar Ver Terminal Solapas Ayuda
npebm@sunny06:~$ cd /boot
npebm@sunny06:/boot$ ls
abi-2.6.22-14-generic          initrd.img-2.6.22-14-generic.bak
config-2.6.22-14-generic      memtest86+.bin
grub                          System.map-2.6.22-14-generic
initrd.img-2.6.22-14-generic  vmlinuz-2.6.22-14-generic
npebm@sunny06:/boot$ cd grub
npebm@sunny06:/boot/grub$ ls
default          installed-version  minix_stagel_5    xfs_stagel_5
device.map       jfs_stagel_5      reiserfs_stagel_5
e2fs_stagel_5    menu.lst          stagel
fat_stagel_5     menu.lst~         stagel2
npebm@sunny06:/boot/grub$
```

3- Creación de usuarios.

Podemos crear usuarios tanto de forma gráfica como desde una terminal. Este trabajo sólo lo podrá realizar el root o un usuario *sudable* (ya se verá que es esto en el apartado 6). Veamos los dos modos:

Forma gráfica:

En el menú *Sistema* elegiremos la opción *Administración* y dentro de ella *Usuarios y grupos*. Aquí nos aparecerán ciertas opciones a cerca de la nueva cuenta creada, como por ejemplo que terminal queremos que utilice por defecto.

Modo texto:

Utilizaremos el comando ***adduser***.

*NOTA: existe otro comando **useradd** pero que no crea el home del usuario*

Una vez tecleado ***adduser usuario*** nos pedirá la contraseña de éste, así como otros datos que podemos rellenar o no, a nuestra elección.

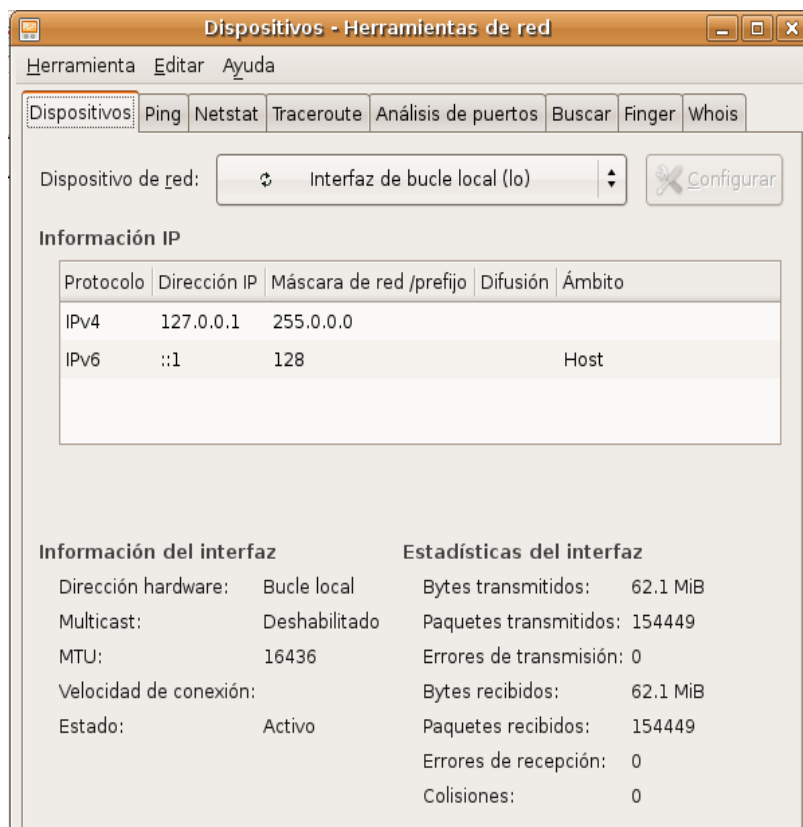
Se puede igualmente eliminar un usuario con el comando ***userdel usuario***, aunque posteriormente habrá que borrar su home: ***rm -r /home/usuario***.

4- Configuración de la red

Para establecer la configuración de nuestra red podemos hacerlo tanto en modo gráfico como desde nuestra terminal.

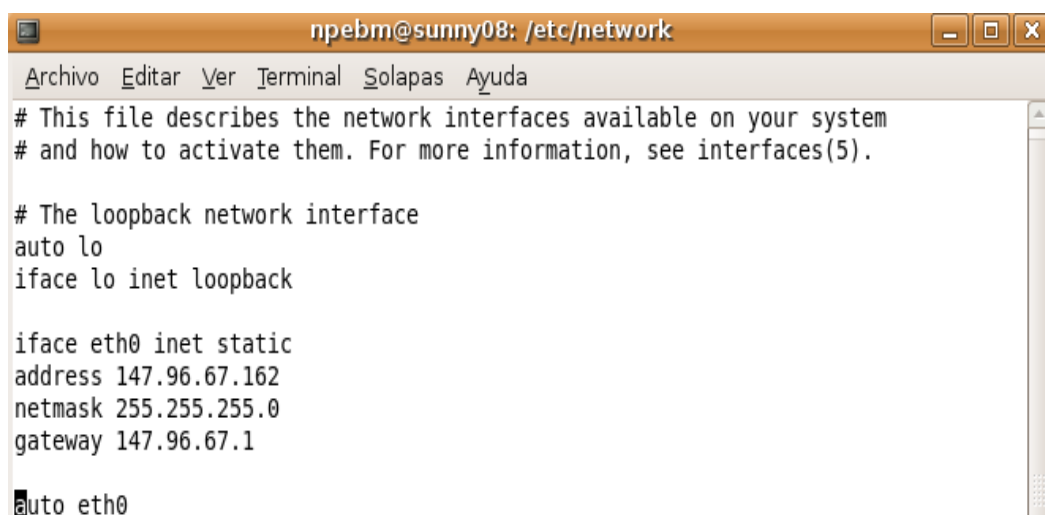
Modo gráfico:

En el menú *Sistema* elegir *Administración* y dentro de este, *Herramientas de red*.

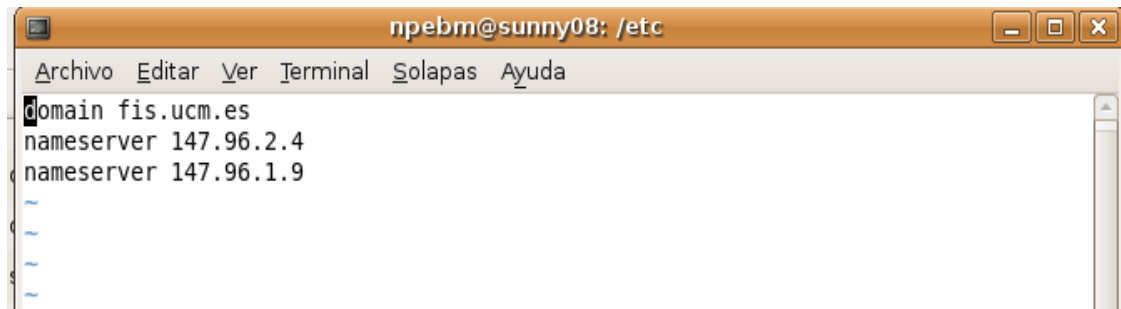


Modo texto:

Accedamos al directorio `cd/etc/network` y modifiquemos el archivo *interfaces*, a través de un editor de texto que veremos a continuación.



Si queremos saber cuales son las *dns* podemos verlo en el archivo `cd /etc/resolv.conf`



```
npebm@sunny08: /etc
Archivo Editar Ver Terminal Solapas Ayuda
domain fis.ucm.es
nameserver 147.96.2.4
nameserver 147.96.1.9
```

Podemos ver también las tarjetas de red con las que contamos, a través del comando *ipconfig* . Aparecerá *eth0 eth1*...tantas como tarjetas tengamos, incluidas las inalámbricas. Con los comandos *ifconfig eth0 down/up* podremos desactivar y activar la tarjeta que deseemos.

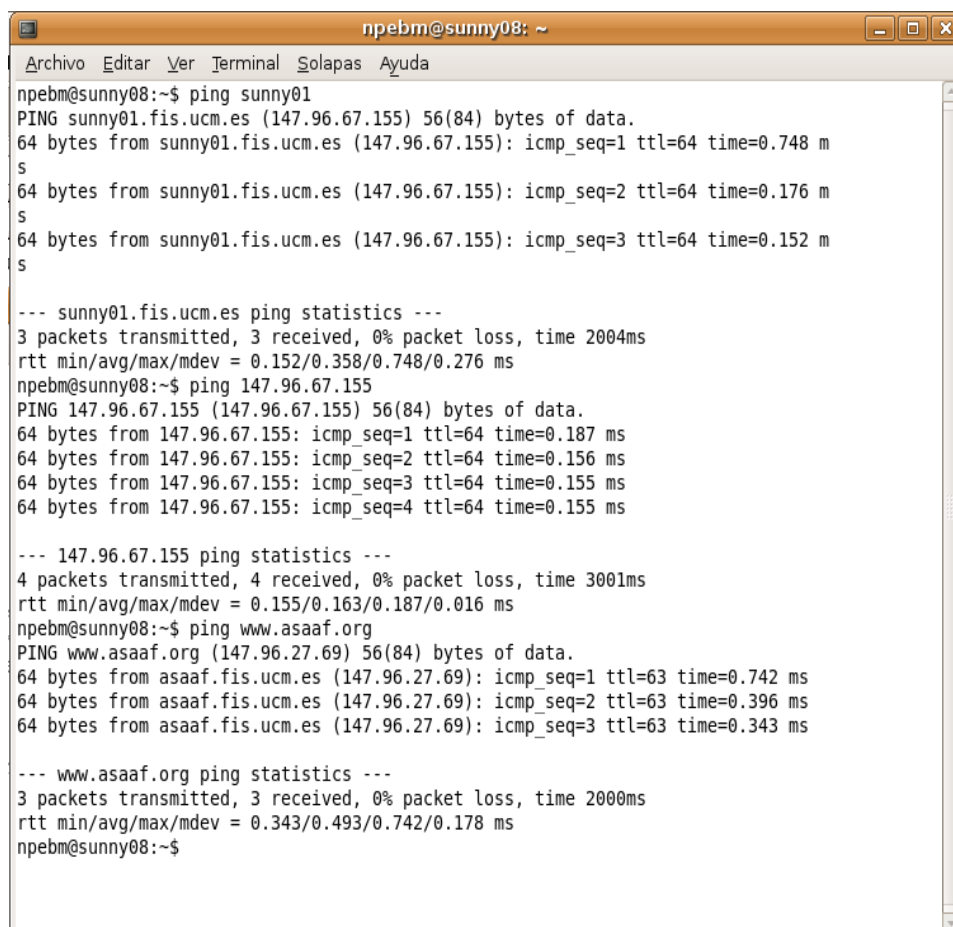
Comando *ping*:

Este comando envía paquetes a otra máquina y espera a que se los devuelva. Podemos ver así si hay algún fallo en la red de una máquina. Para utilizarlo podemos utilizar diferentes sintaxis:

ping sunny01.fis.ucm.es (nombre de la máquina)

ping 147.96.67.155 (IP de la máquina)

ping www.assaaf.org (Con una dirección web)



```
npebm@sunny08: ~
Archivo Editar Ver Terminal Solapas Ayuda
npebm@sunny08:~$ ping sunny01
PING sunny01.fis.ucm.es (147.96.67.155) 56(84) bytes of data.
64 bytes from sunny01.fis.ucm.es (147.96.67.155): icmp_seq=1 ttl=64 time=0.748 m
s
64 bytes from sunny01.fis.ucm.es (147.96.67.155): icmp_seq=2 ttl=64 time=0.176 m
s
64 bytes from sunny01.fis.ucm.es (147.96.67.155): icmp_seq=3 ttl=64 time=0.152 m
s
--- sunny01.fis.ucm.es ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 0.152/0.358/0.748/0.276 ms
npebm@sunny08:~$ ping 147.96.67.155
PING 147.96.67.155 (147.96.67.155) 56(84) bytes of data.
64 bytes from 147.96.67.155: icmp_seq=1 ttl=64 time=0.187 ms
64 bytes from 147.96.67.155: icmp_seq=2 ttl=64 time=0.156 ms
64 bytes from 147.96.67.155: icmp_seq=3 ttl=64 time=0.155 ms
64 bytes from 147.96.67.155: icmp_seq=4 ttl=64 time=0.155 ms
--- 147.96.67.155 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3001ms
rtt min/avg/max/mdev = 0.155/0.163/0.187/0.016 ms
npebm@sunny08:~$ ping www.assaaf.org
PING www.assaaf.org (147.96.27.69) 56(84) bytes of data.
64 bytes from assaaf.fis.ucm.es (147.96.27.69): icmp_seq=1 ttl=63 time=0.742 ms
64 bytes from assaaf.fis.ucm.es (147.96.27.69): icmp_seq=2 ttl=63 time=0.396 ms
64 bytes from assaaf.fis.ucm.es (147.96.27.69): icmp_seq=3 ttl=63 time=0.343 ms
--- www.assaaf.org ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.343/0.493/0.742/0.178 ms
npebm@sunny08:~$
```

Comando *host*:

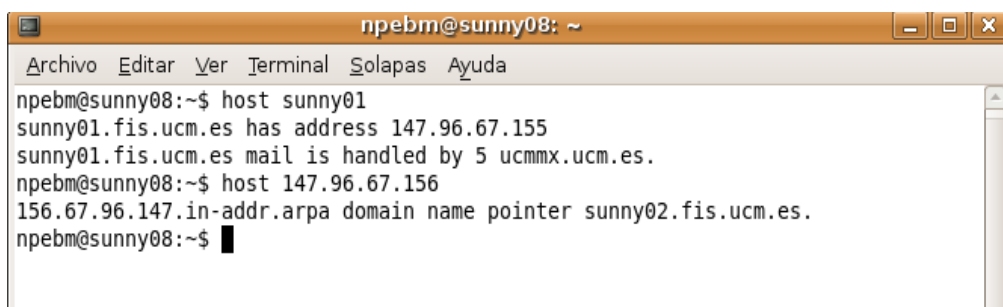
Este comando nos permite conocer a quien pertenece una IP o bien saber que IP corresponde a un nombre conocido. Por ejemplo:

host sunny01.fis.ucm.es

nos devuelve la IP de esa máquina.

host 147.96.22.139

nos dirá que máquina esta conectada con esa IP.



```
npebm@sunny08: ~
Archivo Editar Ver Terminal Solapas Ayuda
npebm@sunny08:~$ host sunny01
sunny01.fis.ucm.es has address 147.96.67.155
sunny01.fis.ucm.es mail is handled by 5 ucmmx.ucm.es.
npebm@sunny08:~$ host 147.96.67.156
156.67.96.147.in-addr.arpa domain name pointer sunny02.fis.ucm.es.
npebm@sunny08:~$
```

Comando *traceroute*

Nos dice por donde pasan los paquetes para llegar de una máquina a otra.

5- Editar archivos de texto

Se utilizan habitualmente tres editores de texto: *gedit*, *emacs* y *vim* (antes conocido como *vi*).

Una vez en el directorio en el que se encuentre el archivo a editar, simplemente tendremos que teclear el nombre del editor a utilizar seguido del nombre del archivo, por ejemplo:

cd /etc

vim hosts

Eso nos abrirá una pantalla con el archivo, sobre la cual podremos ir realizando los cambios necesarios.

5.1- Editor *vim*:

Vi (vim) es un editor de texto para consola. Es el editor de texto tradicional de UNIX, y en muchos sistemas es el único disponible, de manera que es importante saber usarlo, aunque solo sea básicamente.

Para comenzar a editar un archivo deberemos escribir: *vim archivo*, o bien ejecutar *vi*, y luego abrir el archivo con el comando adecuado.

En *vim* existen dos modos de trabajo: un modo de edición y un modo de comandos. Al iniciar el programa, estamos en el modo de comandos. Para ingresar al modo de edición debemos apretar *i*, o bien, *Insert*. Para volver al modo de comandos, utilizamos la tecla *ESC*.

Cuando estemos en el modo de edición, todo lo que ingresemos será texto del archivo. Cuando estemos en el modo comandos, no. A veces lo que escribamos no mostrará ninguna salida inmediata en la pantalla.

Comandos básicos

<i>:e archivo</i>	abre el archivo.
<i>:q</i>	sale del programa, solo si ya se grabaron los cambios.
<i>:q!</i>	sale del programa sin grabar los cambios.
<i>:w</i>	graba el archivo.
<i>:w archivo</i>	graba el archivo con ese nombre (eq. Guardar Como)
<i>:wq</i>	graba el archivo y luego sale del programa.

Teclas de Movimientos

<i>\$</i>	fin de línea.
<i>0</i>	inicio de línea.
<i>b</i>	anterior palabra.
<i>w</i>	próxima palabra.
<i>l</i>	derecha.
<i>h</i>	izquierda.
<i>j</i>	abajo.
<i>k</i>	arriba.
<i>G</i>	fin de archivo.

A la mayoría de estos comandos se les puede agregar un numero al principio. El efecto de este número será el de multiplicar el efecto del comando por el número ingresado. Por ejemplo, *10j* se mueve 10 líneas hacia abajo.

En el caso de *G*, el número que se le agregue antes puede ser el número de línea al cual deseamos ir. Si deseamos ir a la primera línea del archivo, debemos escribir *1G*.

Manejo de Texto

Como en cualquier editor de texto, podemos cortar, copiar y pegar.

<i>dd</i>	corta la línea.
<i>dw</i>	corta la próxima palabra.
<i>d\$</i>	corta hasta el final de la línea.
<i>p</i>	pega lo que se haya cortado o copiado
<i>u</i>	(<i>undo</i>) deshace la ultima acción.
<i>yy</i>	copia la línea.
<i>x</i>	corta el carácter.

Muchos de estos comandos también aceptan un número que los preceda, de tal manera que se pueden seleccionar varios caracteres, palabras o líneas a un mismo tiempo.

Otros

<i>CTRL-g</i>	muestra la línea actual y el total de líneas.
<i>o</i>	agrega una línea debajo de la actual, y entra en modo inserción.
<i>a</i>	se coloca en el carácter siguiente al actual, y en modo inserción.

6- Instalar programas

Sólo el superusuario *root* puede instalar todo tipo de programas. Un usuario normal solo podrá instalar programas en su *home*.

Se denomina usuario *sudable* a aquel que puede hacer lo mismo que el *root* pero a través del comando *sudo*.

NOTA: Ubuntu no trae por defecto un usuario root, si no que el primer usuario que se crea será un usuario con derechos de sudo.

Privilegios:

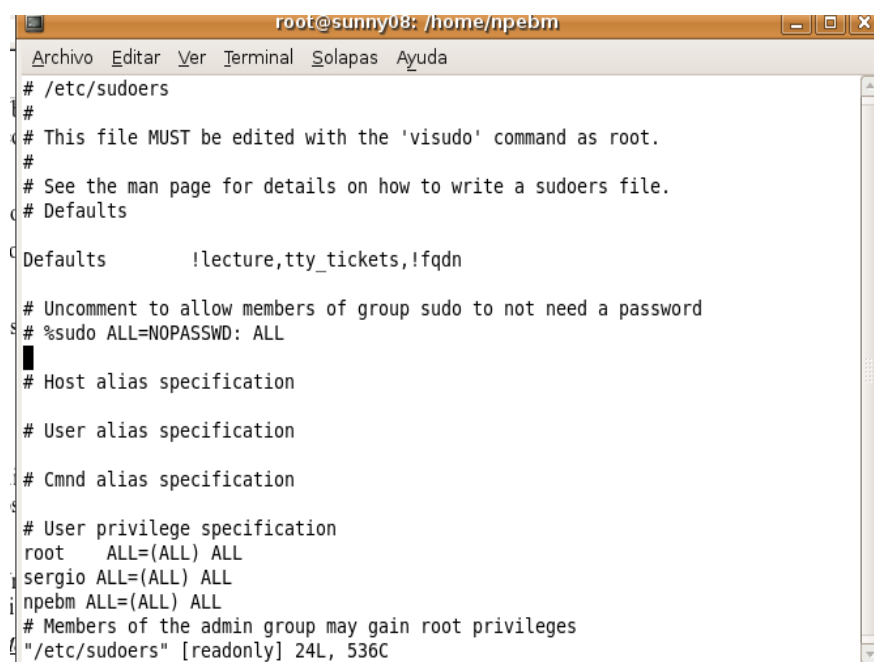
Los privilegios de los usuarios se encuentran reflejados en el archivo */etc/sudoers*.

Podemos modificar este archivo con un editor de los vistos anteriormente, por ejemplo:

```
sudo vim /etc/sudoers
```

Este archivo tiene una línea como la siguiente:

```
#user privilege specification
root    ALL=(ALL) ALL
usuario ALL=(ALL) ALL
```



```

root@sunny08: /home/npebm
Archivo Editar Ver Terminal Solapas Ayuda
# /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the man page for details on how to write a sudoers file.
# Defaults
Defaults    !lecture, tty_tickets, !fqdn
# Uncomment to allow members of group sudo to not need a password
# %sudo ALL=NOPASSWD: ALL
# Host alias specification
# User alias specification
# Cmnd alias specification
# User privilege specification
root    ALL=(ALL) ALL
sergio  ALL=(ALL) ALL
npebm   ALL=(ALL) ALL
# Members of the admin group may gain root privileges
"/etc/sudoers" [readonly] 24L, 536C

```

Esto significa que tanto el root como el usuario de nombre *usuario*, tienen privilegios para todos los comandos.

6.1- *Instalación desde repositorios:*

Un repositorio ofrece programas y actualizaciones. Se pueden descargar programas desde un repositorio tanto en modo gráfico como desde la terminal:

Modo gráfico:

Instalar en el menú *Aplicaciones*, elijamos *Añadir y quitar*

Actualizar en el menú Sistema, elegir Administración, luego Orígenes del software y por ultimo Gestor de actualizaciones.

Modo terminal:

Instalar: `apt-get` realiza una instalación básica

aptitude realiza una actualización más completa.

<i>apt-get install programa</i>	Instala programa
---------------------------------	------------------

<i>apt-get remove programa</i>	Desinstala programa
--------------------------------	---------------------

Con el comando `apt-cache search nombre` encontramos todos los programas que tienen que ver con el *nombre* introducido.

Actualizar: `apt-get update` Busca actualizaciones en los repositorios

<i>apt-get upgrade</i>	Instala actualizaciones bajadas de internet
------------------------	---

apt-get dist-upgrade Actualizamos nuestra distribución

6.2- Descarga y compilación de programas:

Para que solo un usuario pueda utilizar un programa lo descargaremos en su *home*, mientras que si deseamos que todos los usuarios puedan utilizarlo lo guardaremos en el directorio */opt*.

Los programas propios de la distribución que estén en los repositorios se guardaran dentro del directorio */usr*.

Al descargarnos un programa podemos encontrar archivos comprimidos en diferentes formatos:

gzip

.tar

.gzip.tar

Como podemos ver en la imagen siguiente estos archivos suelen aparecer en color rojo.

```
npebm@sunny06: /opt/master/cytoscape2.3
Archivo  Editar  Ver  Terminal  Solapas  Ayuda
npebm@sunny06:/opt/master/cytoscape2.3$ ls
bin          cytoscape.jar          licenses        schema
build        cytoscape.props        linkout.props  src
build.xml    cytoscapeSource-v2.3.tar.gz  manifest       tax_report.txt
config       extra-jars              plugins        testData
cyl.jnlp     help                   props          tools
cytodocs     images                 README.txt     user_docs
cytoscape2.3 lib                    resources
```

Para descomprimir un archivo desde la terminal utilizaremos el comando

`tar -xvzf nombredelarchivo.tar.gz`

También podemos encontrar archivos sources, que vienen escritos en lenguaje C y hay que traducirlos a lenguaje Linux. Para ello usamos la compilación:

`./configure` tiene en cuenta las características de nuestra máquina

`make` compila (en ocasiones también instala)

`make install` Instala

Descarga de archivos o paginas desde Internet:

Para descargar un archivo que este colgado en Internet utilizaremos el comando `wget`, con la siguiente sintaxis:

`wget http://direcciondelaweb`

Si utilizamos `wget -r` nos descargaremos toda la pagina seleccionada.

7- Permisos y propietarios

Como ya se vió en el curso de introducción, al ejecutar el comando `ls -l`, nos aparecen todos los archivos contenidos en el directorio, precedidos por una serie de caracteres que nos indican los tipos de permisos que tiene cada archivo. Veamos un ejemplo:

```
npebm@sunny08:~$ ls -l
total 3852
drwx----- 2 npebm npebm 4096 2008-04-16 21:00 amsn_received
-rw-r--r-- 1 npebm npebm 21161 2008-04-28 16:49 avanzado linux.odt
drwxr-xr-x 10 npebm npebm 4096 2008-03-27 17:57 blimps-3.8
-rw-r--r-- 1 npebm npebm 902399 2008-04-29 16:38 curso introduccion.odt
-rw----- 1 npebm npebm 88807 2008-03-28 20:58 curso_mantenedores.pdf
```

Fijémonos en la carpeta de nombre *blimps-3.8* que tiene permisos *drwxr-xr-x*. Vayamos carácter por carácter viendo su significado:

si es directorio nos aparece *d*

si es archivo nos aparece *-* (como por ejemplo en *curso_mantenedores.pdf*)

si es un enlace (link o icono, acceso directo) aparece *l*

Primer conjunto de tres caracteres *rw**x*: indica cuales son los permisos del propietario

Segundo conjunto de caracteres *r**-x**r*: nos indica los permisos del grupo

Ultimo conjunto de caracteres *r**-x*: los permisos de los que no son usuario principal o grupo (i.e. otros)

Ahora veamos que significan cada una de las letras:

r: lectura

w: escritura

x: ejecución (salen como azulados)

Si deseamos cambiar los permisos de un archivo utilizaremos el comando *chmod*, con la siguiente estructura:

chmod XYZ nombre archivo

donde *X* será un numero que especifique los permisos que le daremos al propietario, *Y* especificara los permisos que le damos al grupo y *Z* especificara los permisos que le damos al resto de usuarios sobre el archivo. Esos números pueden ser:

- 0* no le doy permisos
- 1* permisos de ejecución (*x*)
- 3* permisos de ejecución y escritura (PERO NO LEER!) (*rx*)
- 4* permiso de lectura (*r*)
- 5* permisos de lectura y ejecución (*rx*)
- 6* permiso de lectura y escritura (*rw*)
- 7* permiso de lectura, ejecución y escritura (*rw**x*)

Los más comunes son 1,5,7.

Otra forma de modificar los permisos se basa en la siguiente estructura:

chmod o-x nombrearchivo

El primer carácter, en este caso una *o*, indica de quien queremos cambiar los permisos:

- u* usuario
- g* grupo
- o* otros

El segundo carácter indica si queremos dar más permisos (en ese caso pondremos +) o si queremos quitar permisos (pondremos, como en el ejemplo, un -)

El último carácter indica a qué permisos nos referimos, x: ejecución, w: escritura, r: lectura.

Cambiar propietario:

El comando a utilizar será *chown* con la siguiente sintaxis:

chown nuevopropietario nombrearchivo

Por ejemplo si escribimos en la terminal

chown pepe Archivo

estaremos diciendo que el nuevo propietario del archivo de nombre *Archivo*, será *pepe*.

Podemos cambiar a la vez grupo y propietario, siguiendo la sintaxis siguiente:

chown nuevopropietario:nuevogrupo nombrearchivo

También podemos cambiar solamente el grupo propietario del archivo con el comando *chgrp* y la siguiente sintaxis:

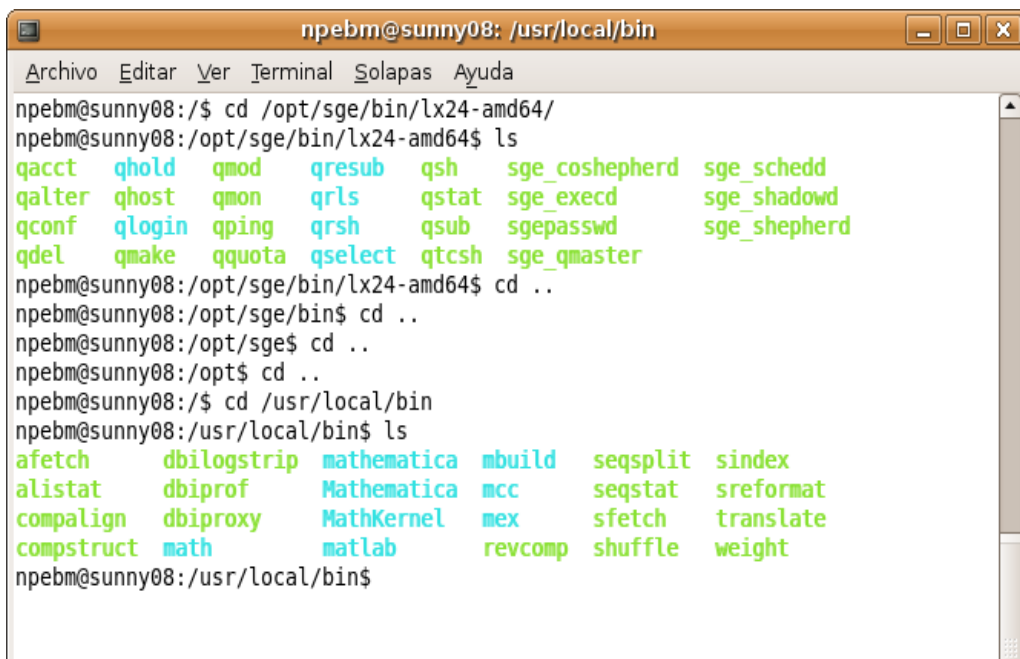
chgrp nuevogrupo nombrearchivo

8- Comandos de la máquina

Los comandos que se ejecutan en la máquina se denominan *Binarios* y se localizan en los siguientes directorios:

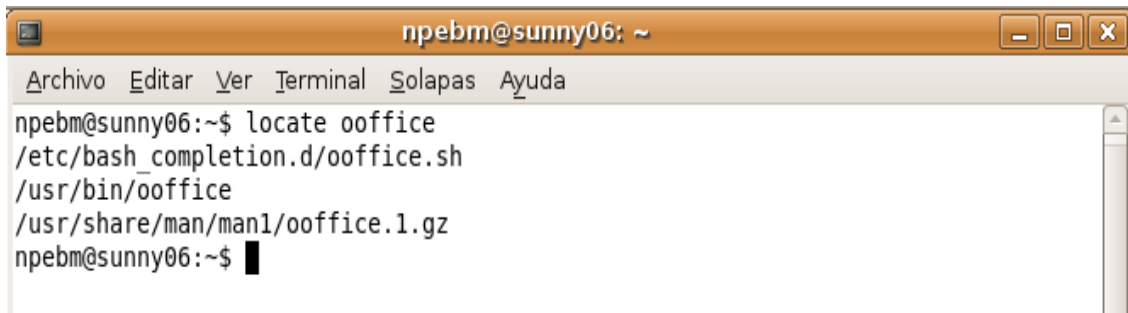
cd /opt/sge/bin

usr/local/bin



```
npebm@sunny08: /usr/local/bin
Archivo  Editar  Ver  Terminal  Solapas  Ayuda
npebm@sunny08:/$ cd /opt/sge/bin/lx24-amd64/
npebm@sunny08:/opt/sge/bin/lx24-amd64$ ls
qacct  qhold  qmod  qresub  qsh  sge_coshepherd  sge_schedd
qalter  qhost  qmon  qrls  qstat  sge_execd  sge_shadowd
qconf  qlogin  qping  qrsh  qsub  sgepasswd  sge_shepherd
qdel  qmake  qquota  qselect  qtcsh  sge_qmaster
npebm@sunny08:/opt/sge/bin/lx24-amd64$ cd ..
npebm@sunny08:/opt/sge/bin$ cd ..
npebm@sunny08:/opt/sge$ cd ..
npebm@sunny08:/opt$ cd ..
npebm@sunny08:/opt/$ cd /usr/local/bin
npebm@sunny08:/usr/local/bin$ ls
afetch  dbilogstrip  mathematica  mbuild  seqsplit  sindex
alistat  dbiprof  Mathematica  mcc  seqstat  sreformat
compalign  dbiproxy  MathKernel  mex  sfetch  translate
compstruct  math  matlab  revcomp  shuffle  weight
npebm@sunny08:/usr/local/bin$
```

Con el comando *locate programa* podemos encontrar donde está ubicado el programa que queremos ejecutar.

A terminal window titled 'npebm@sunny06: ~' with a menu bar (Archivo, Editar, Ver, Terminal, Solapas, Ayuda). The terminal shows the command 'locate ooffice' and its output: '/etc/bash_completion.d/ooffice.sh', '/usr/bin/ooffice', and '/usr/share/man/man1/ooffice.1.gz'. The prompt 'npebm@sunny06:~\$' is followed by a cursor.

Si queremos ejecutar por ejemplo el programa OpenOffice podemos hacerlo de dos modos:

tecleando en la consola directamente *ooffice*

tecleando en consola la ubicación del ejecutable */usr/bin/ooffice*.

Cada usuario en su *home* puede editar el archivo *.bashrc* para que algún programa pueda ser ejecutado simplemente tecleando su nombre en lugar de tener que teclear toda la ruta. Hay que tener en cuenta que el archivo *.bashrc* es un archivo oculto y que para abrirlo y editarlo tendremos que poner en la terminal:

vim.bashrc

(hemos añadido. para que abra el archivo oculto)

Veamos un ejemplo:

Añadamos en nuestro *.bashrc* la siguiente línea:

export GADDIR=/opt/grads/data

De esta forma se ejecutará el programa más cómodamente, y no ha sido necesario más que editar una sola vez el *.bashrc*

*NOTA: Para que sea efectivo el cambio tendremos que cerrar la terminal y abrir una nueva. Si queremos que el cambio afecte a todos los usuarios, la línea la tendremos que añadir al archivo *bash.bashrc* que se sitúa en el directorio etc.*

9- Copiar archivos

Como ya vimos al comienzo del curso, existe un comando que nos sirve para copiar archivos de una ubicación a otra. Ese comando es el *cp*.

La sintaxis a utilizar con este comando es la siguiente:

cp directorio1 directorio2

De esta forma el *directorio1* se habrá copiado en una nueva ubicación dada por *directorio2*. Por ejemplo:

cp /etc/vim/vimrc /home/npebm/.vimrc

Así el archivo *vimrc* que se situaba dentro de */etc/vim* lo habremos copiado al *home* del usuario *npebm* con el nombre *.vimrc*

Si lo que queremos copiar es una carpeta completa y no solo un archivo utilizaremos el comando

cp -r

Comando scp:

Este comando es análogo al *cp* pero se utiliza para copiar archivos desde un ordenador remoto, es decir, si por ejemplo hemos instalado un programa externo a las librerías en uno de los ordenadores del aula y queremos trasladarlo al resto. La sintaxis para utilizar este comando es la siguiente:

scp archivoorigen usuarioremoto@máquina remota:destino

Por ejemplo, si en nuestro *home* tenemos el archivo *StarOffice.odt* y queremos trasladárselo a otro usuario de nombre *usuario* que se sitúa en la máquina *sunny02*, haríamos:

scp /home/npebm/StarOffice.odt usuario@sunny02.fis.ucm.es:/home/usuario

De este modo habremos trasladado el archivo que el usuario *npebm* tenía en su *home*, al *home* del usuario de nombre *usuario*, que estaba en la máquina *sunny02*

Al igual que con el comando *cp*, si lo que queremos copiar de forma remota son carpetas en lugar de archivos utilizaremos el comando *scp -r*. Hay que tener cuidado al utilizar este comando porque si ponemos por ejemplo:

scp -r /home/npebm/StarOffice.odt usuario@sunny02.fis.ucm.es:/home/usuario/

se creará una carpeta vacía entre el usuario y el archivo

10- Escritorio remoto. Vnc (Visual Network Computing)

En el caso de tener varios ordenadores conectados en red en nuestra casa u oficina puede resultar muy interesante poder controlar y acceder a algunos de ellos sin tener que estar sentados físicamente delante.

Esto es posible gracias al *escritorio remoto*. El ordenador al que queremos acceder hace de servidor. El ordenador desde el que accedemos es el cliente de escritorio remoto. Tanto para establecer el servidor como el cliente de escritorio remoto necesitamos configurar el sistema para permitir y realizar conexiones.

Vayamos por partes. Primero el servidor, éste puede tener instalado cualquier sistema operativo que permita aceptar conexiones remotas.

En Ubuntu es tan fácil como permitir que los usuarios remotos vean mi escritorio desde *Sistema >> Preferencias >> Escritorio Remoto*

Una vez que tenemos el servidor configurado pasamos a ver el cliente de escritorio remoto que necesitaremos en el equipo desde el que queremos conectarnos.

Si tenemos instalado Ubuntu u otra distribución de GNU/Linux podemos emplear el cliente de *Terminal Server*, que encontraremos en el menú *Aplicaciones >> Internet >> Cliente de Terminal Server*

Configuramos la pantalla que nos aparecerá con los datos adecuados para nuestro servidor de escritorio remoto y hacemos clic en *Conectar*. Si todo ha ido bien aparecerá una ventana con el escritorio del otro ordenador, por la que podremos movernos y interactuar como si estuviésemos sentados delante.

El programa que más se utiliza para esta tarea se denomina *VNC*, siglas de *Virtual Network Computing*, y lo que hace es permitir a un ordenador remoto usar su propia pantalla, teclado y ratón como si fueran los propios. Vamos a ver detenidamente, paso a paso, el proceso de puesta en marcha de este programa.

Al instalar el programa, cuando nos ofrezca seleccionar los componentes a instalar, debemos diferenciar entre si lo hacemos en el cliente o en el servidor, en el caso del servidor, instalamos las dos opciones, pues va a hacer tanto las veces de servidor como cliente de sí mismo o de otra máquina. Para el ordenador cliente, en cambio, tan sólo necesitamos instalar la opción *VNC Viewer*, pues vamos a usarlo para visualizar el escritorio de otro ordenador.

Una vez que ya tenemos instaladas las dos versiones, el servidor y el visor de datos, procedemos a preparar el servicio. En el ordenador que hace las veces de servidor, debemos arrancar el programa «*Run VNC Server*». Nada más arrancar, se nos presentará una ventana en la que haremos una configuración rápida de las propiedades del servidor. Los parámetros y opciones que vienen por defecto suelen ser perfectamente válidas, y tan sólo estás obligado a introducir la contraseña de acceso al servidor, pues sin ella no te dejará arrancar. Una vez introducida, ya puedes pinchar en OK, con lo que se cerrará la ventana, y te aparecerá un icono en la zona inferior derecha de la pantalla.

Ahora debes pasar al ordenador que hará las funciones de cliente. Aquí debes arrancar el programa «*Run VNC Viewer*». La ventana que se te presenta en primer lugar te solicita el nombre de la máquina donde te quieres conectar. Esta denominación la puedes entregar tanto como nombre de red, como directamente como dirección TCP/IP. A continuación te preguntará la contraseña y deberás indicar la que antes escribiste.

Ahora se te presentará una nueva ventana que tendrá en su interior el contenido del escritorio del ordenador remoto, que podrás manejar desde tu ordenador, con tu propio teclado y ratón. De hecho, todas las acciones que realices en remoto, se mostrarán tanto en la ventana local como en la pantalla remota.

Vnc es un programa de software libre basado en una estructura cliente-servidor el cual nos permite tomar el control de un ordenador servidor remotamente a través de un ordenador cliente. Lo podemos encontrar dentro del menú *Aplicaciones*, seleccionando *Internet* y posteriormente *Visor de escritorios remoto*. Sólo podemos ver máquinas en modo servidor que tengan instaladas las siguientes aplicaciones:

vnc-java

x11vnc es el cliente de “terminal server”

La aplicación *x11vnc* sirve para poner la máquina en modo servidor.

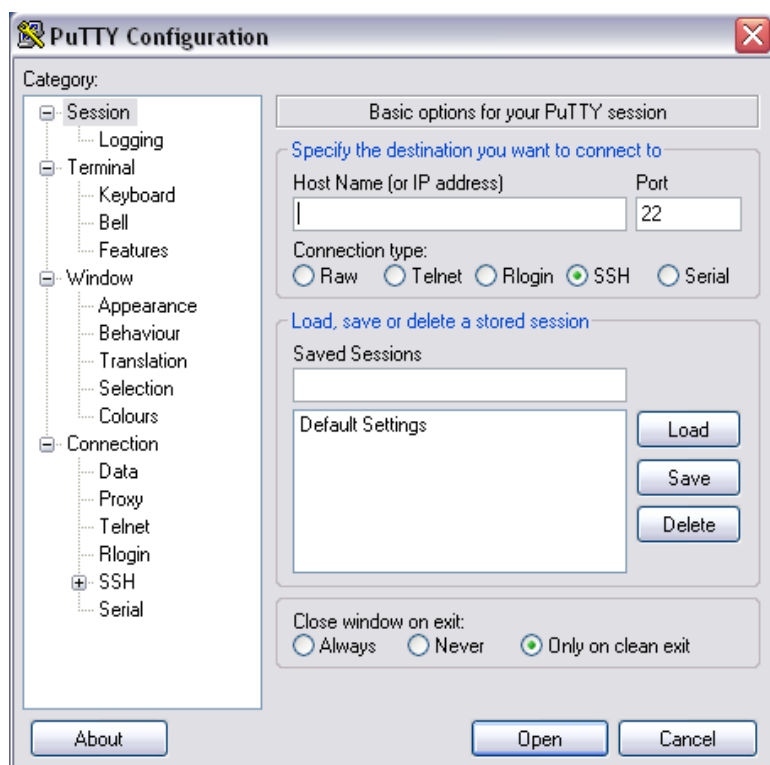
x11vnc -storepasswd Si queremos poner una contraseña

x11vnc -viewonly Para poder utilizar el ratón desde fuera

11- Conectarse a una máquina con Linux desde Windows.

Si deseamos desde nuestro ordenador en casa con Windows, conectarnos remotamente a otra máquina que tenga Linux, podemos utilizar las siguientes aplicaciones:

putty.exe no necesita instalación. Nos proporciona una terminal de la máquina a la que nos hayamos conectado



winssh funciona como el protocolo ssh

winscp permite ver los escritorios tanto de nuestro equipo como del equipo al que nos hemos conectado.

Existe la aplicación *xming* que usa simultáneamente los protocolos *ssh -x* y *xdmcp*, y que permite usar los programas de *Windows* desde *Linux*

Además tenemos a nuestra disposición:

cygwin paquete que funciona en Windows y sirve para correr aplicaciones de Linux

wine paquete para Linux que permite correr programas de Windows.

12- Servidor de páginas Web

Un *servidor Web* no es más que un programa que ejecuta de forma continua en un ordenador (también se utiliza el término para referirse al ordenador que lo ejecuta), manteniéndose a la espera de peticiones por parte de un cliente (un navegador de Internet) y que contesta a estas peticiones de forma adecuada, sirviendo una página Web que será mostrada en el navegador o mostrando el mensaje correspondiente si se detectó algún error.

Instalar un servidor Web en nuestro PC nos permitirá, entre otras cosas, poder montar nuestra propia página Web sin necesidad de contratar hosting, probar nuestros desarrollos en local, acceder a los ficheros de nuestro ordenador desde un PC remoto (aunque para esto existen otras alternativas, como utilizar un servidor FTP) o utilizar alguno de los programas basados en Web. Uno de los servidores Web más populares del mercado, y el más utilizado actualmente, es **Apache**, de código abierto y gratuito, disponible para Windows y GNU/Linux, entre otros.

13 - Scripts

Los Scripts son archivos con extensión *.sh* que se ejecutan en terminal (*bash* o *sh*)

Para ejecutar un *script* no hay más que teclear en la terminal: *./nombre del script*
 Si no estuviéramos en terminal *bash* teclearíamos entonces: *sh ./nombre del script*

Todos los scripts deben empezar con un conjunto de caracteres conocido como *sha bang* y que es: *#!/bin/bash*

A la hora de escribir un script podemos crear variables que posteriormente podremos llamar dentro del mismo script. Por ejemplo:

Con *A=7* estaríamos definiendo la variable *A* con el valor 7

Con *\$A* estaríamos llamando a la variable *A* cuyo valor ya había sido definido.

Obviamente podemos hacer que *A* fuera un vector o cualquier otro conjunto de caracteres u operaciones más complejo, pero ésto se ve en otro curso, dedicado simplemente a la creación de scripts.

El comando *read* es capaz de leer lo que el usuario introduce por el teclado:

<i>read -n</i>	nos devuelve el número de caracteres
<i>read -p</i>	nos devuelve una frase
<i>read -d</i>	sirve para delimitar

Podemos también detectar errores en nuestro script si lo iniciamos con la siguiente sintaxis:

#!/bin/bash -x|-v

Al haber añadido *-x* se nos mostraran instrucciones antes de ejecutar el script y al añadir *-v* nos pondrá las variables que hemos definido, seguidas de su valor.

Los comandos más utilizados en la creación de scripts son:

if equivale a un “si”. La sintaxis es *if condición*. Es necesario cerrarlo con un *fi*
else

elif equivale a un *else+if*. No es necesario cerrarlo

case se utiliza cuando queremos reflejar varios casos de operación en nuestro script. Es necesario cerrarlo con un *esac*.

while equivale a un “mientras que”, y sirve para indicarle al script, por ejemplo, cuando debe dejar de ejecutarse, o cuando cambiar el tipo de ejecución

do;done

14 - Demonios

Los *daemons* (o demonios) no son más que un proceso que se ejecuta en segundo plano. Estos demonios ejecutan diferentes funciones y proporcionan ciertos servicios, pero sin la interacción del usuario; son procesos de los que no "notamos" su ejecución.

Los demonios pueden ser iniciados al arrancar el sistema, al entrar en un nivel de ejecución determinado, o simplemente cuando nosotros los iniciemos. Veremos de qué modo podemos controlar nosotros mismos los demonios y cómo podemos hacer que se inicien automáticamente.

Control de demonios

Los programas que ejecutamos como demonios pueden estar ubicados en cualquier parte del disco, pero tienen un punto en común: todos utilizan un script para ser iniciados/parados, y estos scripts se encuentran en el directorio: */etc/init.d/*



```
npebm@sunny08:~$ cd /etc/init.d
bash: cd: /etc/init.d: No existe el fichero ó directorio
npebm@sunny08:~$ cd /etc/init.d
npebm@sunny08:/etc/init.d$ ls
acpid          kdm-kde4      rc.local
acpi-support   keyboard-setup rcS
alsa-utils     killprocs     readahead
anacron        klogd         readahead-desktop
apparmor       laptop-mode   README
appport        libpam-foreground
atd            linux-restricted-modules-common
avahi-daemon   lm-sensors   reboot
binfmt-support loopback      rmnologin
bluetooth      makedev       rsync
bootclean      module-init-tools
bootlogd       mountall-bootclean.sh
bootmisc.sh    mountall.sh   screen-cleanup
brltty         mountdevsubfs.sh
checkfs.sh     mountkernfs.sh
checkroot.sh   mountnfs-bootclean.sh
console-screen.sh
console-setup  mtab.sh
cron           networking
cryptdisks     nfs-common    sendsig
udev
```

La sintaxis habitual para iniciar/parar demonios es:

/etc/init.d/<nombre_demonio> start (para iniciar el demonio)
/etc/init.d/<nombre_demonio> stop (para detenerlo)

Será conveniente conocer los *run levels*:

/etc/inittab

0	halt	<i>etc/rc0</i>
3	multisistema	<i>etc/rc3</i>
5	entorno gráfico	<i>etc/rc5</i>
6	reboot	<i>etc/rc6</i>

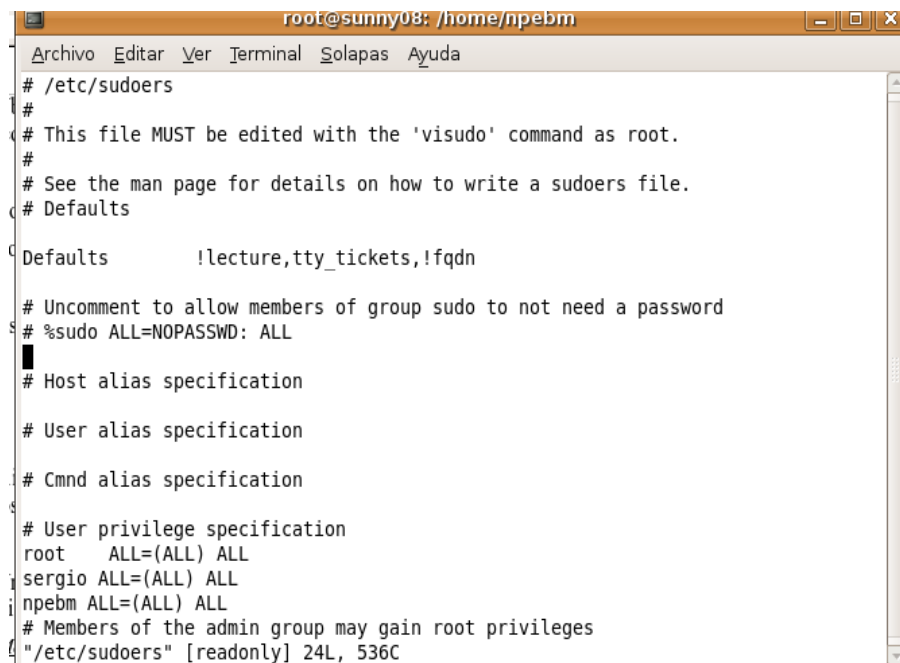
Scripts propios

Aparte de los demonios del sistema y los demonios de determinadas aplicaciones, nosotros también podemos crear nuestros scripts para iniciar/parar/reiniciar los programas que nosotros queramos poner como demonios. Únicamente hemos de crear un script que siga la siguiente sintaxis:

```
#!/bin/sh
case "$1" in
start)
# código para iniciar el demonio/programa
;;
stop)
# código para parar el demonio/programa
;;
restart)
# código para reiniciar el demonio/programa
;;
esac
```

Esto nos dice que cuando ejecutemos el script y le pasemos el parámetro *start*, nos iniciará el demonio, cuando le pasemos el parámetro *stop* lo detendrá, y cuando le pasemos el parámetro *restart* lo reiniciará. Normalmente sólo *root* tiene permisos para iniciar/parar los servicios, pero todo depende de los permisos que se le hayan puesto al script.

Podemos crear un usuario que tenga privilegios para la creación de scripts y de modo que no le sea requerida la contraseña. Para ello tendremos que añadir una línea en el archivo *sudoers* como ya vimos que se hacía en el apartado 6 de este curso, cuando hablamos de privilegios para instalar programas. Recordemos el aspecto de ese archivo:



```
root@sunny08: /home/npebm
Archivo Editar Ver Terminal Solapas Ayuda
# /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the man page for details on how to write a sudoers file.
# Defaults
Defaults        !lecture, tty_tickets, !fqdn
# Uncomment to allow members of group sudo to not need a password
# %sudo ALL=NOPASSWD: ALL
# Host alias specification
# User alias specification
# Cmnd alias specification
# User privilege specification
root    ALL=(ALL) ALL
sergio  ALL=(ALL) ALL
npebm   ALL=(ALL) ALL
# Members of the admin group may gain root privileges
"/etc/sudoers" [readonly] 24L, 536C
```


CURSO AVANZADO DE LINUX

Tenemos dos opciones:

Añadir una nueva línea en *# User privilege specification*

script ALL=NOPASSWD: /etc/init.d/script

Añadir una línea en *# Cmnd alias specification*

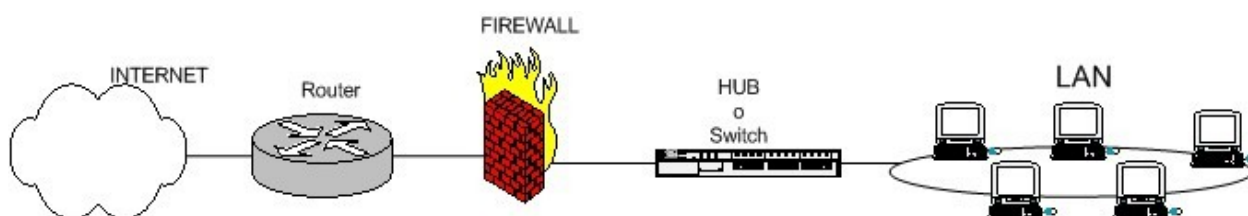
IPW: /etc/init.d/script1 , /etc/init.d/script2

script ALL:NONPASSWD:PW

15- Firewall e IPtables

Un firewall es un dispositivo que filtra el tráfico entre redes, como mínimo dos. El firewall puede ser un dispositivo físico o un software sobre un sistema operativo. En general debemos verlo como una caja con dos o más interfaces de red en la que se establecen una reglas de filtrado con las que se decide si una conexión determinada puede establecerse o no.

Esa sería la definición genérica, hoy en día un firewall es un hardware específico con un sistema operativo o una IOS que filtra el tráfico TCP/UDP/ICMP/..IP y decide si un paquete pasa, se modifica, se convierte o se descarta. Para que un firewall entre redes funcione como tal debe tener al menos dos tarjetas de red. Esta sería la tipología clásica de un firewall:



Los firewalls se pueden usar en cualquier red. Es habitual tenerlos como protección de internet en las empresas, aunque ahí también suelen tener una doble función: controlar los accesos externos hacia dentro y también los internos hacia el exterior; esto último se hace con el firewall o frecuentemente con un proxy (que también utilizan reglas, aunque de más alto nivel).

Sea el tipo de firewall que sea, generalmente no tendrá mas que un conjunto de reglas en las que se examina el origen y destino de los paquetes del protocolo tcp/ip. En cuanto a protocolos es probable que sean capaces de filtrar muchos tipos de ellos, no solo los tcp, también los udp, los icmp, los gre y otros protocolos vinculados a vpns. Este podría ser (en pseudo-lenguaje) un el conjunto de reglas de un firewall como el del gráfico:

Política por defecto ACEPTAR.

CURSO AVANZADO DE LINUX

Todo lo que venga de la red local al firewall ACEPTAR

Todo lo que venga de la ip de mi casa al puerto tcp 22 ACEPTAR

Todo lo que venga de la ip de casa del jefe al puerto tcp 1723 ACEPTAR

Todo lo que venga de hora.rediris.es al puerto udo 123 ACEPTAR

Todo lo que venga de la red local y vaya al exterior ENMASCARAR

Todo lo que venga del exterior al puerto tcp 1 al 1024 DENEGAR

Todo lo que venga del exterior al puerto tcp 3389 DENEGAR

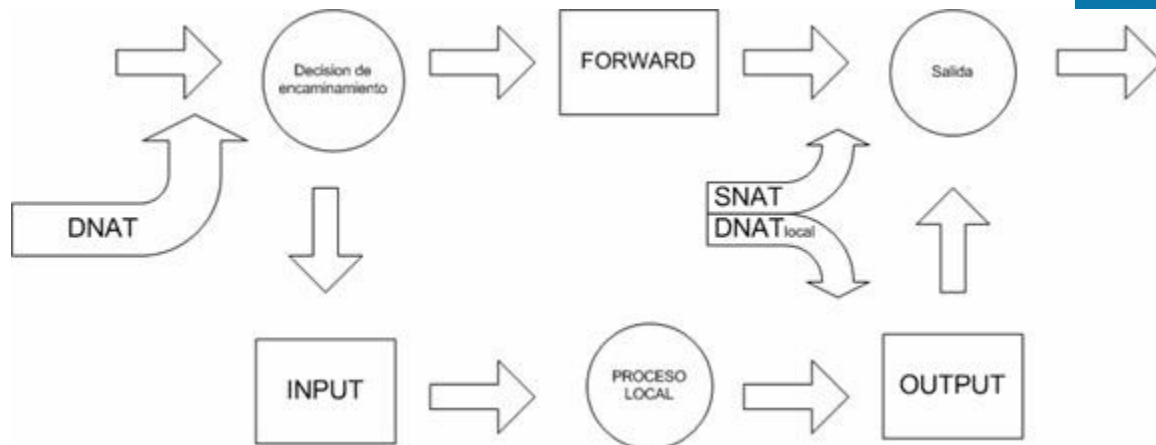
Todo lo que venga del exterior al puerto udp 1 al 1024 DENEGAR

En definitiva lo que se hace es:

- Habilita el acceso a puertos de administración a determinadas IPs privilegiadas
- Enmascara el tráfico de la red local hacia el exterior (NAT, una petición de un pc de la LAN sale al exterior con la ip pública), para poder salir a internet
- Deniega el acceso desde el exterior a puertos de administración y a todo lo que este entre 1 y 1024.

IPtables es un sistema de firewall vinculado al kernel de Linux que se ha extendido enormemente a partir del kernel 2.4 de este sistema operativo. Al igual que el anterior sistema ipchains, un firewall de iptables no es como un servidor que lo iniciamos o detenemos o que se pueda caer por un error de programación (esto es una pequeña mentira, ha tenido alguna vulnerabilidad que permite DoS, pero nunca tendrá tanto peligro como las aplicaciones que escuchan en determinado puerto TCP): iptables esta integrado con el kernel, es parte del sistema operativo. Un firewall de iptables no es sino un simple script de shell en el que se van ejecutando las reglas de firewall.

Como se ve en el siguiente gráfico, básicamente se mira si el paquete esta destinado a la propia máquina o si va a otra. Para los paquetes (o datagramas, según el protocolo) que van a la propia máquina se aplican las reglas INPUT y OUTPUT, y para filtrar paquetes que van a otras redes o maquinas se aplican simplemente reglas FORWARD. INPUT, OUTPUT y FORWARD son los tres tipos de reglas de filtrado. Pero antes de aplicar esas reglas es posible aplicar reglas de NAT: éstas se usan para hacer redirecciones de puertos o cambios en las IPs de origen y destino.



15.1- Firewall para nuestra máquina

Vamos a ver a continuación un ejemplo de como crear con Iptables un firewall para nuestra maquina. Lo unico que tendríamos que hacer seria un script de shell en el que se vayan aplicando reglas. Los scripts de IPTables pueden tener este aspecto:

```
#!/bin/sh
## SCRIPT de IPTABLES – ejemplo del manual de iptables
## Ejemplo de script para proteger la propia máquina
echo -n Aplicando Reglas de Firewall...
## FLUSH de reglas
iptables -F
iptables -X
iptables -Z
iptables -t nat -F
## Establecemos politica por defecto
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT

## Empezamos a filtrar

# El localhost se deja (por ejemplo conexiones locales a mysql)
/sbin/iptables -A INPUT -i lo -j ACCEPT
# A nuestra IP le dejamos todo
iptables -A INPUT -s 195.65.34.234 -j ACCEPT
# A un colega le dejamos entrar al mysql para que mantenga la BBDD
iptables -A INPUT -s 231.45.134.23 -p tcp --dport 3306 -j ACCEPT
```

CURSO AVANZADO DE LINUX

```
# A un diseñador le dejamos usar el FTP
iptables -A INPUT -s 80.37.45.194 -p tcp --dport 20:21 -j ACCEPT
```

```
# El puerto 80 de www debe estar abierto, es un servidor web.
```

```
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

```
# Y el resto, lo cerramos
```

```
iptables -A INPUT -p tcp --dport 20:21 -j DROP
```

```
iptables -A INPUT -p tcp --dport 3306 -j DROP
```

```
iptables -A INPUT -p tcp --dport 22 -j DROP
```

```
iptables -A INPUT -p tcp --dport 10000 -j DROP
```

```
echo " OK . Verifique que lo que se aplica con: iptables -L -n"
```

```
# Fin del script
```

El script es simple, con unas pocas reglas con las que cerramos puertos al público a los que no tienen porque tener acceso, salvo el 80. Pero si nos damos cuenta no se filtra el UDP ni el ICMP. Es muy probable que el sistema tenga algún puerto udp abierto, y además peligroso como el SNMP. Para evitar problemas, al final del script podemos añadir la orden de cerrar el rango de puertos del 1 al 1024, que son los reservados tanto para tcp como udp.

```
...
```

```
...
```

```
# El puerto 80 de www debe estar abierto, es un servidor web.
```

```
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

```
# Cerramos rango de los puertos privilegiados. Cuidado con este tipo de
# barreras, antes hay que abrir a los que si tienen acceso.
```

```
iptables -A INPUT -p tcp --dport 1:1024
```

```
iptables -A INPUT -p udp --dport 1:1024
```

```
# Cerramos otros puertos que estan abiertos
```

```
iptables -A INPUT -p tcp --dport 3306 -j DROP
```

```
iptables -A INPUT -p tcp --dport 10000 -j DROP
```

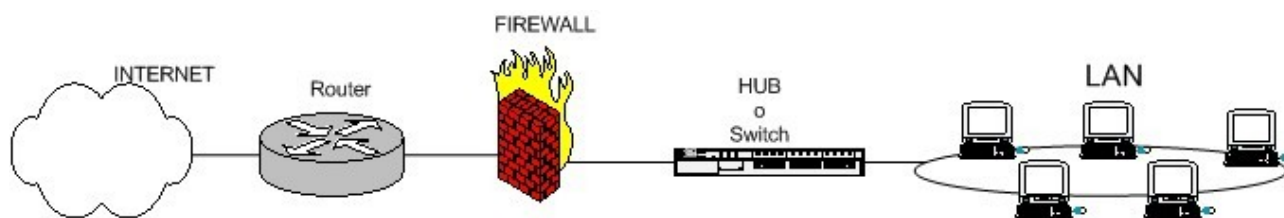
```
iptables -A INPUT -p udp --dport 10000 -j DROP
```

```
echo " OK . Verifique que lo que se aplica con: iptables -L -n"
```

```
# Fin del script
```

15.2 Firewall para una red local con salida a internet

Ahora vamos a ver una configuración de firewall iptables para el típico caso de red local que necesita salida a internet.



¿Qué es lo que hace falta? Obviamente, una regla que haga NAT hacia fuera (enmascaramiento en iptables), con lo que se haría dos veces NAT en el firewall y en el router. Entre el router y el firewall lo normal es que haya una red privada (192.168.1.1 y 192.168.1.2 por ejemplo), aunque dependiendo de las necesidades puede que los dos tengan IP pública. El router se supone que hace un NAT completo hacia dentro (quizá salvo puerto 23), o sea que desde el exterior no se llega al router si no que de forma transparente se "choca" contra el firewall. Lo normal en este tipo de firewalls es poner la política por defecto de FORWARD en denegar (DROP).

Veamos como sería este firewall:

```
#!/bin/sh
## SCRIPT de IPTABLES – ejemplo del manual de iptables
## Ejemplo de script para firewall entre red-local e internet
echo -n Aplicando Reglas de Firewall...
## FLUSH de reglas
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

## Establecemos politica por defecto
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT

## Empezamos a filtrar
## Nota: eth0 es el interfaz conectado al router y eth1 a la LAN
# El localhost se deja (por ejemplo conexiones locales a mysql)
/sbin/iptables -A INPUT -i lo -j ACCEPT
# Al firewall tenemos acceso desde la red local
iptables -A INPUT -s 192.168.10.0/24 -i eth1 -j ACCEPT

# Ahora hacemos enmascaramiento de la red local
# y activamos el BIT DE FORWARDING (imprescindible!!!!)
```

CURSO AVANZADO DE LINUX

```
iptables -t nat -A POSTROUTING -s 192.168.10.0/24 -o eth0 -j MASQUERADE
# Con esto permitimos hacer forward de paquetes en el firewall, o sea
# que otras máquinas puedan salir a través del firewall.
echo 1 > /proc/sys/net/ipv4/ip_forward
```

```
## Y ahora cerramos los accesos indeseados del exterior:
# Nota: 0.0.0.0/0 significa: cualquier red
# Cerramos el rango de puerto bien conocido
iptables -A INPUT -s 0.0.0.0/0 -p tcp -dport 1:1024 -j DROP
iptables -A INPUT -s 0.0.0.0/0 -p udp -dport 1:1024 -j DROP
# Cerramos un puerto de gestión: webmin
iptables -A INPUT -s 0.0.0.0/0 -p tcp -dport 10000 -j DROP
echo " OK . Verifique que lo que se aplica con: iptables -L -n"
```

15.3- Un ejemplo más de Firewall

Otra forma de crear un firewall es mediante los archivos /etc/host.deny y /etc/host.allow
En el primero especificaremos los puertos y las máquinas que no dejaremos acceso, en el segundo a las que sí se lo dejaremos.

Ejemplo host.deny

ALL:ALL Este es el modo pánico, cerramos todas las conexiones entrantes
SSH:ALL Cerraremos todas las conexiones entrantes por el puerto ssh.
ALL:ip Cerraremos la entrada a la ip especificada a todas las conexiones.

Si combinamos las ordenes de host.allow con host.deny podemos crear un firewall a medida.

15.4- Logueo en red sin contraseña

Muchas veces debemos ejecutar órdenes remotas, mediante scripts o demonios y para ello necesitamos una forma de loguearnos por ssh sin tener que dar el password.
Por cuestiones de seguridad, para estas cosas crearemos un usuario, puede hacerse como root pero es más inseguro.

Es necesario eliminar cualquier fingerprint que pueda llegar a tener registrado la máquina cliente que apunte a la máquina servidor utilizando root u otro usuario distinto al que va a generar el túnel automáticamente. Para esto, en el cliente ejecutamos:

```
/root/.ssh
#vim known_hosts
```

buscamos la entrada que apunta al servidor y la eliminamos. O bien eliminamos todo el archivo.

Ahora necesitamos crear el par de claves RSA pública/privada para el cliente .
Con éste fin utilizamos el comando ssh-keygen del paquete SSH .

Para éste caso en particular he dejado por defecto el nombre de archivo de las claves a generar (id_rsa e id_rsa.pub) y no he utilizamos una frase o hash alguno:

```
ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_rsa.
Your public key has been saved in id_rsa.pub.
The key fingerprint is:
a6:63:a7:3e:7d:17:42:83:b2:85:79:ef:44:e8:89:67
```

Ahora dentro de la carpeta /root/.ssh tendremos los dos archivos que contienen el par de claves. id_rsa e id_rsa.pub

Para que el servidor no requiera autenticación del equipo cliente es necesario incluir la clave pública del cliente (el contenido del archivo /root/.ssh/id_rsa.pub que acabamos de crear) dentro del archivo de claves autorizadas para el usuario del servidor con el que queremos entrar (que se encuentra /home/usuario/.ssh/authorized_keys).

A partir de éste momento el cliente es capaz de iniciar una sesión segura en el servidor logeandose como un usuario sin necesidad de ingresar contraseña alguna.

Como ya hemos visto el comando con el cual logrará esto es:

```
ssh servidor -l usuario
```

16- Copias de Seguridad

Es muy importante guardar una copia de nuestros archivos periódicamente para evitar pérdidas de información.

Vamos a ver varias formas de hacerlo: por una parte haremos backups de ficheros y directorios, esto nos permitirá poder restaurar archivos o directorios de manera individual y por otra haremos una copia nos permitirá recuperar nuestro sistema en caso de urgencia.

En ambos casos, el directorio de destino puede ser local o remoto.

16.1 Backups de directorios y ficheros

Se trata de hacer un backup de todos los archivos de nuestro usuario. A demás, mantendremos un histórico de backups, haremos una copia diaria (lunes, martes, miércoles, etc .), cada lunes machacaremos la copia del lunes anterior y así con todos los días.

El procedimiento es sencillo, elegiremos los directorios y ficheros de los que queremos hacer el backup, y los copiaremos.

```
cp -rp /home /media/sdb/backup/domingo
```

Es totalmente funcional, pero la verdad es que no es eficiente, el problema está en que cada lunes nos hará una copia completa de todos los archivos que tenemos en el directorio, cuando lo que seria deseable es que sólo copiara los modificados o los creados des del último backup.

CURSO AVANZADO DE LINUX

No hay problema, tenemos otra instrucción que nos ofrece esta función.

```
rsync -altgvbp /home/banyut /media/sdb5/backup/domingo
```

Puede ser interesante hacerlo en formato comprimido, para ello generaremos un *.tar*

He creado la carpeta */backups* en el raíz que es donde se guardará el script y los archivos *tar.gz* a transmitir.

El script lo llamo *backup.sh* y debe tener permisos de ejecución (*chmod 744*).

A continuación se detalla el código de *backup.sh* :

```
#!/bin/bash
#####
# Script de backup automatico #
#####
# ----- Asignacion de variables -----
# Esta variable es igual al nombre del dia actual
# (se utilizara mas adelante para darle nombre al archivo)
DIA=`date +%A`
#Directorios a backupear
DIRECTORIOS= "/usr/local/bin /usr/local/sbin /etc"
#----- Realizacion del backup -----
cd /backup
logger "*** Realizando Backup correspondiente al dia $DIA ***"
if [ -f $DIA.tar.gz ];then
rm $DIA.tar.gz
fi
tar -czf $DIA.tar.gz $DIRECTORIOS &> /dev/null
logger "*** Realizando transferencia ***"
scp $DIA.tar.gz ipdestino
logger "*** Transferencia del backup finalizada ***"
```

La variable *\$DIA* graba el nombre en inglés del día actual. Con esto se logra que cada backup dure una semana ya que el script pisa un archivo con el mismo nombre (en realidad lo borra primero y regenera después).

La variable *\$DIRECTORIOS* contiene la ruta de los directorios a backupear separadas por un espacio.

Las variables *\$DEST* y *\$PUERTO* indican parámetros para realizar la transferencia.

Por último se genera una tarea en CRON para ejecutar el script una vez por día:

```
# backup automatico
30 20 * * * root /bin/sh /backup/backup.sh
```


16.2- Backup de recuperación

En este tipo de backups, lo que buscamos es realizar una copia tal cual de nuestro disco.

Utilizaremos el comando *dd*

El comando *dd* (*duplicate disk*) es un comando bastante útil para transferir datos desde un dispositivo/archivo hacia un dispositivo/archivo/etc.

La sintaxis básica del comando es la siguiente:

```
dd if=origen of=destino
```

donde *if* significa "*input file*", es decir, lo que querés copiar y *of* significa "*output file*", o sea, el archivo destino (donde se van a copiar los datos); *origen* y *destino* pueden ser dispositivos (lectora de CD, diskettera, etc.), archivos, etc.

Copiando Diskettes:

Primero insertamos el diskette origen y escribimos lo siguiente en una consola:

```
dd if=/dev/fd0 of=~/.diskette.img
```

después insertamos el diskette destino (en blanco) y escribimos lo siguiente:

```
dd if=~/.diskette.img of=/dev/fd0
```

nos queda eliminar la "imagen" que creamos y listo...

```
rm -f ~/.diskette.img
```

NOTA: El ~ significa "*tu directorio home*", es similar a escribir **\$HOME**

Manejo de errores durante la copia: Es posible que durante la copia o duplicación de un diskette se encuentren errores en la superficie del mismo. Para evitar que este error nos impida copiar los datos "buenos" del disco podemos hacer lo siguiente:

```
dd conv=noerror if=/dev/fd0 of=~/.imagen_disco_con_errores.img
```

NOTA: la opción *noerror* hace que se continúe con la copia aunque se produzcan errores de lectura

Haciendo imágenes ISO de un CD:

La forma mas fácil y efectiva de crear nuestras "imágenes" de CD es la siguiente:

```
dd if=/dev/cdrom of=micd.iso
```

El comando *dd* también sirve para copiar particiones o discos completos unos sobre otros.

Básicamente podemos decir que mediante *dd* podemos "*clonar*" particiones o nuestro disco rígido completo. Para hacer esto hacé lo siguiente:

```
dd if=/hdx a of=/hdyb      (copia una partición en otra)
dd if=/hdx of=/hdy        (copia de un disco duro en otro)
```

donde: *x*: disco rígido origen, *y*: disco rígido destino, *a*: partición origen, *b*: partición destino.

NOTA: Es necesario que sepas como se definen los discos y particiones en Linux.

16.3- Recomendaciones

* Tener mucho cuidado con lo que haces porque los datos de la partición o disco destino serán destruidos por completo.

* Tener en cuenta también que la partición o disco destino debe ser igual en tamaño (o en todo caso mayor) que la partición o disco origen.

* Es conveniente hacer una copia de seguridad de los datos importantes y tener a mano un disco de arranque de linux por si acaso.

Por último: Tener presente que el tamaño de la imagen resultante va ser exactamente el mismo que el del dispositivo original. Es decir: **dd** te guarda también el espacio no utilizado.

Se puede redirigir la salida con una tubería (pipe) y comprimirlo con *gzip*, *bzip*, pero aún así vas a necesitar bastante espacio libre para poder guardar las imágenes que generes.

17- Anexo: Comandos de interes

A continuación mostramos una tabla con los comandos mas utilizados en Linux, con una breve descripción de su uso:

COMANDO	DESCRIPCIÓN
<i>apropos palabra</i>	Ver comandos relacionados con <i>palabra</i>
<i>which comando</i>	Ver la ruta completa de <i>comando</i>
<i>time comando</i>	Medir cuanto tarda <i>comando</i>
<i>time cat</i>	Iniciar cronómetro. <i>Ctrl-d</i> para detenerlo.
<i>nice info</i>	Lanzar comando <i>info</i> con prioridad baja
<i>renice 19 -p \$\$</i>	Darle prioridad baja al shell (guión). Usar para tareas no interactivas
dir navegación	
<i>cd -</i>	Volver al directorio anterior
<i>cd</i>	Ir al directorio personal (home)
<i>(cd dir && comando)</i>	Ir a dir, ejecutar comando y volver al directorio inicial
<i>pushd .</i>	Guardar el directorio actual en la pila para luego, poder hacer <i>popd</i> y volver al mismo
búsquedas de archivo	
<i>alias l='ls -l --color=auto'</i>	Listado de directorio rápido
<i>ls -lrt</i>	Listar archivos por fecha. Ver también <i>newest</i>
<i>ls /usr/bin pr -T9 -W\$COLUMNS</i>	Imprimir 9 columnas en ancho de la terminal
<i>find -name '*.ch' xargs grep -E 'expre'</i>	Buscar 'expre' en este directorio y subdirectorios.
<i>find -type f -print0 xargs -r0 grep -F 'ejemplo'</i>	Buscar 'ejemplo' en todos los archivos regulares en este directorio y subdirectorios
<i>find -maxdepth 1 -type f xargs grep -F 'ejemplo'</i>	

	Buscar 'ejemplo' en todos los archivos regulares de este directorio
<code>find -maxdepth 1 -type d \ while read dir; do echo \$dir; echo cmd2; done</code>	Procesar cada elemento con muchos comandos (con un bucle while)
<code>find -type f ! -perm -444</code>	Hallar archivos sin permiso general de lectura (util para sedes web)
<code>find -type d ! -perm -111</code>	Hallar directorios sin permiso general de acceso (util para sedes web)
<code>locate -r 'file[^\]*\.txt'</code>	Buscar nombres en índice en cache. Este re es igual a <code>glob *file*.txt</code>
<code>look referencia</code>	Búsqueda rápida (ordenada) de prefijo en diccionario
<code>grep --color referencia /usr/share/dict/palabras</code>	Resaltar ocurrencias de expresión regular en diccionario
archivos	
<code>gpg -c file</code>	Encriptar archivo
<code>gpg file.gpg</code>	Desencriptar archivo
<code>tar -c dir/ bzip2 > dir.tar.bz2</code>	Crear archivo compacto de dir/
<code>bzip2 -dc dir.tar.bz2 tar -x</code>	Extraer archivo compacto (usar gzip en vez de bzip2 para archivos tar.gz)
<code>tar -c dir/ gzip gpg -c ssh user@remoto 'dd of=dir.tar.gz.gpg'</code>	Crear compactado encriptado de dir/ en equipo remoto
<code>find dir/ -name '*.txt' tar -c --files-from=- bzip2 > dir_txt.tar.bz2</code>	Crear compactado de subconjunto de dir/ y subdirectorios
<code>find dir/ -name '*.txt' xargs cp -a --target-directory=dir_txt/ --parents</code>	Copiar subconjunto de dir/ y subdirectorios
<code>(tar -c /dire/de/copiame) (cd /este/dir/ && tar -x -p)</code>	Copiar (con permisos) directorio copiame/ a directorio / este/dir/
<code>(cd /dire/de/copiame && tar -c .) (cd /este/dir/ && tar -x -p)</code>	Copiar (con permisos) contenido del directorio copiame/ a directorio /este/dir/
<code>(tar -c /dire/de/copiame) ssh -C user@remoto 'cd /este/dir/ && tar -x -p'</code>	Copiar (con permisos) directorio copiame/ a directorio remoto /este/dir/
<code>dd bs=1M if=/dev/hda gzip ssh user@remoto 'dd of=hda.gz'</code>	Respaldo de disco duro en equipo remoto
rsync (Usar la opción --dry-run para probarlo)	
<code>rsync -P rsync://rsync.servidor.com/ruta/a/archivo archivo</code>	Obtener solo <i>diffs</i> . Repetir muchas veces para descargas conflictivas
<code>rsync --bwlmit=1000 desdearchivo alarchivo</code>	Copia local con taza límite. P
<code>rsync -az -e ssh --delete ~/public_html/ remoto.com:~/public_html'</code>	Espejo de sede web (usando compresión y encriptado)
<code>rsync -auz -e ssh remote:/dir/ . && rsync -auz -e ssh . remote:/dir/</code>	Sincronizando directorio actual con uno remoto

wget (herramienta de descargas multiuso)	
<code>(cd cmdline && wget -nd -pHEKk http://www.pixelbeat.org/cmdline.html)</code>	Guardar en directorio actual una versión navegable de una página web
<code>wget -c http://www.ejemplo.com/largo.archivo</code>	Retomar descarga de un archivo parcialmente descargado
<code>wget -r -nd -np -l1 -A '*.jpg' http://www.ejemplo.com/</code>	Descargar una serie de archivos en el directorio actual
<code>wget ftp://remoto/archivo[1-9].iso/</code>	FTP permite globalizaciones directas
<code>wget -q -O- http://www.pixelbeat.org/timeline.html grep 'a href' headP</code>	Procesando directamente la salida
<code>echo 'wget url' at 01:00</code>	Descargar la url a 1AM al directorio en que esté
<code>wget --limit-rate=20k url</code>	Hacer descargas de baja prioridad (en este caso, no exceder los 20KB/s)
<code>wget -nv --spider --force-html -i bookmarks.html</code>	Revisando los enlaces de una página
<code>wget --mirror http://www.ejemplo.com/</code>	Actualizar eficientemente una copia local de una página web (útil si usamos cron)
redes	
<code>ethtool interface</code>	Listar estado de interfase
<code>ip link show</code>	Listar interfases
<code>ip link set dev eth0 name wan</code>	Renombrar eth0 a wan
<code>ip addr add 1.2.3.4/24 brd + dev eth0</code>	Agregar ip y máscara (255.255.255.0)
<code>ip link set dev interface up</code>	Subir (o bajar) la interfase
<code>ip route add default via 1.2.3.254</code>	Establecer 1.2.3.254 como valor por omisión para la puerta de enlace.
<code>tc qdisc add dev lo root handle 1:0 netem delay 20msec</code>	Agregarle 20ms de espera al dispositivo de retorno (para hacer pruebas)
<code>tc qdisc del dev lo root</code>	Quitar la espera agregada antes.
<code>host pixelbeat.org</code>	Obtener la dirección ip para el dominio o al revés
<code>hostname -i</code>	Obtener la dirección ip local (equivale al anfitrión `hostname`)
<code>netstat -tupl</code>	Listar los servicios de internet de un sistema
<code>netstat -tup</code>	Listar las conexiones activas de/hacia un sistema
windows (nota samba es el paquete que permite todos estos comandos de redes de windows)	
<code>smbtree</code>	Hallar equipos windows. Ver también find smb
<code>nmblookup -A 1.2.3.4</code>	Hallar el nombre (netbios) de windows asociado con la dirección ip
<code>smbclient -L windows_box</code>	Listar archivos compartidos en equipos windows o servidor samba
<code>mount -t smbfs -o fnask=666,guest //windows_box/share /mnt/share</code>	Montar un directorio compartido

<i>calendario</i>	
<i>cal -3</i>	Mostrar calendario
<i>cal 9 1752</i>	Mostrar calendario para mes y año determinado
<i>date -d fri</i>	Que día cae este viernes. Ver también <i>day</i>
<i>date --date='25 Dec' +%A</i>	¿En que día cae la Navidad, este año?
<i>date --date '1970-01-01 UTC 1234567890 seconds'</i>	Convertir total de segundos desde la época a una fecha
<i>TZ=':America/Los_Angeles' date</i>	¿Que hora es en la Costa Oeste de EEUU (usar <i>tzselect</i> para hallar TZ)
<i>echo "mail -s 'tomar el tren' P@draigBrady.com < /dev/null" at 17:45</i>	Recordatorio por email
<i>echo "DISPLAY=\$DISPLAY xmessage cooker" at "NOW + 30 minutes"</i>	Recordatorio emergente
<i>locales</i>	
<i>printf "%d\n" 1234</i>	Imprimir numero agrupado por miles de acuerdo a su locale
<i>BLOCK_SIZE=\1 ls -l</i>	pedir que ls agrupe por miles de acuerdo a su locale
<i>echo "Yo vivo en `locale territory`"</i>	Extraer información de la base de datos del locale
<i>LANG=en_IE.utf8 locale int_prefix</i>	Buscar información de locale para determinado país.
<i>locale cut -d= -f1 xargs locale -kc less</i>	Listar campos en base de datos del locale
<i>información del sistema</i>	
<i>hdparm -i /dev/hda</i>	Ver informe sobre partición hda
<i>hdparm -tT /dev/hda</i>	Hacer una prueba de velocidad de lectura en partición hda
<i>badblocks -s /dev/hda</i>	Hallar bloques ilegibles en partición hda
<i>mount column -t</i>	Ver particiones montadas en el sistema (y alinear la salida)
<i>cat /proc/partitions</i>	Ver todas las particiones registradas en el sistema
<i>grep MemTotal /proc/meminfo</i>	Ver el total de RAM que registra el sistema
<i>grep "model name" /proc/cpuinfo</i>	Ver informe de CPU(s)
<i>lspci -tv</i>	Ver informe de PCI
<i>lsusb -tv</i>	Ver informe de USB

Estos apuntes son el resultado de las clases impartidas en el aula SUN, rogamos a nuestros alumnos que aporten todo el material extra que sea posible con el fin de mejorarlas y ampliarlas.