COMANDOS BÁSICOS

`cat > archivo`
Crea archivo de textolinea a linea hasta que se pulsa Ctrl-D
`cd $HOME ó cd ~`
Cambia a nuestro directorio home
`cp`
Copia archivos
`dmesg`
Muestra mensajes del kernel
`du`
Muestra el espacio ocupado en disco
`echo`
Muestra una linea de texto
`fdisk` (tambien `cfdisk` en algunos sistemas)
Programa de formateo del disco duro
`grep`
Muestra las lineas que coindicen con un patron
`hostname`
Muestra o establece el nombre del sistema
`kill`
Manda señales a procesos (p.ej. mata procesos)
`id usuario`
Muestra info sobre usuario
`less`
Muestra texto linea a linea
`ln`
Crea enlaces entre ficheros
`locate, which, whereis` ó
`´find ./ -name archivo´`
Busca un determinado archivo en el sistema
`login`
Comienza una sesión en el sistema
`lspci`
Muestra información sobre el bus PCI
`mknod`
Crea archivos especiales tipo bloque o carácter (/dev)
`more`
Muestra texto pantalla a pantalla
`mount/umount`
Monta/desmonta un sistema de ficheros
`mv`
Mueve o renombra archivos
`passwd`
Cambia la clave de usuario
`pwd`
Muestra el directorio en el que estamos
`reset`
re-inicializa la terminal
`rm -r dir`
Borra archivos (y directorios recursivamente)
`sort archivo`
Ordena archivo de texto
`uname`
Muestra información del sistema
`wc`
Cuenta los bytes, las palabras y lineas de un archivo.
`who`
Muestra los usuarios conectados actualmente al sistema
`^zz`
Sale de pág. man

VISIONADO DE ARCHIVOS

`cat`	vuelca archivo a pantalla
`more`	vuelca archivo a pantalla con pausa en cada una
`tail`	muestra el final de un archivo
`head`	muestra el comienzo de un archivo
`less`	similar a more pero con navegación linea a linea

COMUNICACIÓN ENTRE USUARIOS

`mail juan,paco < mensaje.txt`
manda un correo con el archivo mensaje.txt a los usuarios del sistema juan y paco
`write usuario [tty]`
manda mensaje a otro usuario que esté usando el sistema actualmente
`wall [archivo]`
manda mensaje a todos los usuarios conectados en ese momento
`mesg [y ó n]`
activa o desactiva la capacidad de enviar mensajes a otros usuarios

Nota: `ls -l /dev/tty#` muestra los permisos de una consola determinada. `mesg` en realidad simplemente cambia los permisos de escritura de la consola. Se podría hacer de otra manera con un `chmod`.

PROCESOS

`ps -eax`	muestra todos los procesos
`jobs`	similar a ps pero muestra solo trabajos
`bg proc`	manda proc a segundo plano
`fg proc`	trae proc a primer plano
`comando &`	ejecuta comando en segundo plano
`kill id`	mata proceso
`kill %numjob`	mata trabajo

`$ prog1; prog2; prog3`
(se ejecutan los 3 comandos uno despues de otro)
`$ prog1; prog2; prog3&`
(prog3 se ejecuta en segundo plano)
`$ (prog1; prog2; prog3)&`
(todos se ejecutan en segundo plano)
`$ (prog1; prog2; prog3) > cosa.txt &`
(redirigimos la salida a cosa.txt)

Estado *zombie*: es cuando se mata al proceso padre y los procesos hijos siguen ejecutándose

COMUNICACIÓN ENTRE PROCESOS

`ls | sort -r`
lista invertida de archivos
`who | wc -l`
cuenta las lineas de la salida de who
`who | tee listausr.txt`
crea archivo con la salida de who
`ps | sort | tee estado.txt`
ordena salida de ps y crea archivo estado.txt con la salida
`who | grep aitor | wc -l`
cuenta las lineas en donde aparece aitor como usuario.

PLANIFICACIÓN DE PROCESOS

`at hora fecha [+incremento]`
ejecuta comandos a una hora determinada
`batch`
similar a at pero espera a que se descongestione el sistema las ordenes dadas
`nice -nn prog`
modifica el nivel de prioridad de *prog*
`nohup`
deja el programa ejecutándose aunque salgamos del sistema
`cron`
demonio encargado de realizar tareas programadas
`crontab arch`
maneja la lista de tareas programas del usuario
Ejemplos:
`at now +3 hours comando`
Ejecuta comando dentro de 3 horas

Formato archivo crontab:

x1	x2	x3	x4	x5	x6
minutos (0-59)	horas (0-23)	dias (1-31)	mes (1-12)	dia semana (0-6)	comando

Nota: en `X5` los días empiezan por domingo (0=sunday). Se pueden utilizar también sus nombres en inglés.

`now`	ahora
`noon`	mediodia
`midnight`	medianoche
`teatime`	4 de la tarde

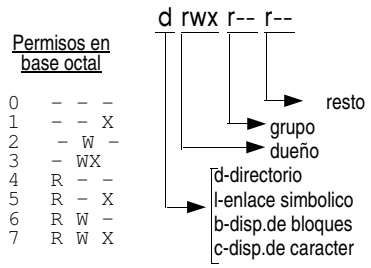
`at 10 wed comando`
Ejecuta comando el próximo miércoles a las 10
`at 12 20 Mar 14 comando`
Ejecuta comando el 14 de Marzo a las 12:20 horas
`at 8pm tomorrow comando`
(equivalente a `at 20 00 tomorrow comando`)
Ejecuta comando mañana a las 8 de la tarde
`0 12 15 * * comando`
Ejecuta comando cada día 15 a las 12:00
`0,30 9-17 9,14 3 * comando`
Ejecuta comando cada media hora, entre las 9 y las 5 de la tarde, los días 9 y 14 de Marzo.

OPERACIONES CON ÓRDENES

;	Concatenar órdenes (ej: `$who; date`)
&&	Solo ejecuta la segunda orden en caso de que la primera resulte ok
||	Solo se ejecuta la segunda orden en caso de error de la primera
>	Envia la salida de un comando a otro
<	Toma la salida de un comando como entrada del primero.

PERMISOS

Comandos relacionados: `chmod`, `chgrp`, `umask`



Ej.: `chmod u=u+rw, g=r, o=r arch`

(cambia permisos de *arch*. **Añade** a los que tenga el usuario, lectura y escritura, y establece los del grupo y los otros a solo lectura. También podría hacerse así:

Ej. equivalente: `chmod 644 arch`

VI

INSERCION DE TEXTO

i ó [INSERT]	Insertar texto
O	Inserta linea sobre de la actual
o	Inserta linea bajo la actual

DESPLAZAMIENTO

:set number	Numera las lineas
G	Ir al final de archivo
gg	Ir a comienzo de archivo
num G	Va a linea *num*
num k	Sube *num* lineas
$	Va al final de la linea actual
0	Va al comienzo de linea actual
e	Va al final de la siguiente palabra
w	Va al comienzo de la siguiente palabra
b	Va al comienzo de la palabra anterior
Ctrl-u	Sube 12 lineas
Ctrl-d	Baja 12 lineas

ELIMINACION Y RECUPERACION DE TEXTO

x	Elimina carácter hacia delante
X	Elimina carácter hacia atrás
dw	Borra la palabra sobre cursor
d$	Borra hasta el final de la linea actual
u	Deshace la última acción
Ctrl-R	Rehace la última acción

CAMBIO DE TEXTO

r *letra*	Cambia carácter actual por *letra*
R	Modo reemplazar (igual que pulsar dos veces [INSERT])

BUSQUEDA Y SUSTITUCION DE TEXTO

f *letra*	Busca *letra* en la linea
F *letra*	Busca hacia atrás *letra* en la linea
;	Va a siguiente búsqueda de letra
/*palabra*	Busca *palabra* en texto
?*palabra*	Busca hacia atrás *palabra* entexto
n	Se situa en la siguiente palabra encontrada
N	Se situa en la anterior palabra encontrada
:s/cadenavieja/cadenanueva	Sustituye cadena *vieja* por *nueva* en la linea actual
:1,$s/cadenavieja/cadenanueva	Sustituye cadena viaja por nueva en todo el fichero

CORTAR, COPIAR Y PEGAR TEXTO

y *num*	Selecciona texto.
y)	Selecciona hasta final de la frase
p	Pegar texto seleccionado detrás de la linea actual
d *num*	Borra *num* lineas desde cursor (se almacenan en un buffer)
a5dd	Borra 5 lineas y las pasa al buffer a
ap	Pega el buffer a

OTROS

[ESC]	Modo comando
:wq	Escribe arch y sale
:w *nomfich*	Escribe archivo
^ZZ	Igual que :wq
:q!	Sale de VI descartando cambios
:r *archivo*	Inserta arch en posición del cursor
:sh	Salir momentaneamente al shell. exit para volver al editor

BASH - SHELL SCRIPTS

REDIRECCIÓN

stdout a un fichero (crea un fichero que contiene lo que se vería en pantalla si se escribiese el comando)
```
$ ls -l > fichero.txt
```
stderr a un fichero (crea un fichero que contiene los errores que se verían en pantalla si se escribiese el comando)
```
$ grep da * 2> errores-de-grep.txt
```
stdout a stderr (esto hará que la salida de stdout se escriba en el mismo descriptor de fichero que stderr)
```
$ grep da * 1>&2
```
stderr a stdout (esto hará que la salida stderr de un programa se escriba en el mismo descriptor de fichero que stdout)
```
$ grep da * 2>&1
```
stderr y stdout a un fichero (esto colocará toda la salida de un programa en un fichero. A veces esto es conveniente en las entradas del cron, si quiere que un comando se ejecute en absoluto silencio)
```
$ rm -f $(find / -name core) &> /dev/null
```

TUBERÍAS

Las tuberías permiten utilizar la salida de un programa como la entrada de otro:

```
$ who | wc -l
```

Cuenta las lineas de la salida del comando who (lo que nos daría la cantidad de usuarios conectados en ese momento)

VARIABLES

Una variable de bash puede contener un número, un carácter o una cadena de caracteres. No se necesita declarar una variable, se creará sólo con asignarle un valor a su referencia.

```
$ ./script arg1 arg2 ... arg9
```

`$0` variable especial que contiene el propio nombre del script que se ha ejecutado
`$*` variable especial que contiene el nº de argumentos pasados con el script
`$1` Primer argumento
`$9` Ultimo argumento

Para leer una variable desde el teclado:
```
read VARIABLE
```
Para asignar un valor a una variable:
```
$VARIABLE = valor
```

Truco: Para esperar que el usuario pulse una tecla definir un `read` sin variable.
Notas:
Hay formas de pasar más de nueve argumentos a un script.
Como norma las variables se ponen en mayúscula.

Ejemplo:

```
#!/bin/bash
CAD="Hola Mundo"
echo $CAD
```

Variables locales
Pueden crearse variables locales utilizando la palabra clave *local*. Ejemplo:

```
#!/bin/bash
HOLA=Hola
function hola {
  local HOLA=Mundo
  echo $HOLA
}
echo $HOLA
hola
echo $HOLA
```

ALGUNAS VARIABLES

`HOME`	Nombre de la ruta del directorio de usuario.
`SHELL`	Nombre de la shell actual
`MAIL`	Ruta del buzón del usuario
`LOGNAME`	Nombre del login actual
`HISTNAME`	Nombre del archivo del historial
`PATH`	Rutas en las que el shell busca los comandos
`PS1`	Prompt o indicador.
`TERM`	Tipo de terminal
`TZ`	Zona horaria.

ESTRUCTURAS CONDICIONALES

if...then...else...fi

Las estructuras condicionales permiten decidir si se realiza o no una acción; esta decisión se toma evaluando una expresión.

La base de las construcciones 'if' es esta:

```
if [expresión];
then
código si 'expresión' es verdadera
fi
```

Ejemplos:

```
#!/bin/bash
# Ejemplo basico de if...then
if [ "petete" = "petete" ]; then
  echo la expresión es verdadera
fi
```

```
#!/bin/bash
# Ejemplo basico de if...then...else
if [ "petete" = "petete" ]; then
  echo la expresión es verdadera
else
  echo la expresión es falsa
fi
```

```
#!/bin/bash
# Ejemplo de condicionales con
variables
T1="petete"
T2="peteto"
if [ "$T1" = "$T2" ]; then
  echo expresión verdadera
else
  echo expresión falsa
fi
```

Los bucles for; while y until

- El bucle `for` le permite iterar sobre una serie de "palabras" contenidas dentro de una cadena.
- El bucle `while` ejecuta un trozo de código si la expresión de control es verdadera, y solo se para cuando es falsa.
- El bucle `until` es casi idéntico al bucle while, excepto que el código se ejecuta mientras la expresión de control se evalúe como falsa.
- `break`: sale de un bucle for, while, until o select.

Ejemplos:

```
#!/bin/bash
for i in $ (ls); do
  echo item: $i
done
```

```
#!/bin/bash
for i in 'seq 1 10';
do
  echo $i
done
```

```
#!/bin/bash
CONTADOR=0
while [ $CONTADOR -lt 10 ]; do
  echo El contador es $CONTADOR
  let CONTADOR=CONTADOR+1
done
```

```
#!/bin/bash
CONTADOR=20
until [ $CONTADOR -lt 10 ]; do
  echo CONTADOR $CONTADOR
  let CONTADOR-=1
done
```

FUNCIONES

Se pueden utilizar funciones para agrupar trozos de código de una manera más lógica. Ejemplo:

```
#!/bin/bash
function salir {
  exit
}
function hola {
  echo "Hola"
}
hola
salir
```

FUNCIONES (CONT)

Ejemplo de funciones con parámetros:

```
#!/bin/bash
function salir {
  exit
}
function e {
  echo $1
}
e hola
e Mundo
```

INTERFACES DE USUARIO

Utilizando `select` para hacer menús sencillos:

```
#!/bin/bash
OPCIONES="Hola Salir"
select opt in $OPCIONES; do
if [ "$opt" = "Salir" ]; then
  echo done
  exit
elif [ "$opt" = "Hola" ]; then
  echo Hola Mundo
else
  clear
  echo opción errónea
fi
done
```

Utilizando `case` para hacer menús sencillos:

```
#!/bin/bash
echo Teclee 1, 2 o 3. S para
salir:
read LETRA
case $LETRA in
  1) echo "Has tecleado 1" ;;
  2) echo "Has tecleado 2" ;;
  3) echo "Has tecleado 3";;
  S) echo Saliendo...
  *) echo Opcion erronea ;;
  clear
    exit ;;
esac
```

Nota: Por cada opción del `case` se acaba con un doble punto y coma.

DEPURACION DE SCRIPTS

Para depurar la ejecución de un script:

```
$ sh -x script
```

SHELL BASH

Fichero de inicio de sesión
```
$HOME/.bash.profile
```
Fichero de personalización de entorno
```
$HOME/.bashrc
```
Fichero de fin de sesión
```
$HOME/.bash.logout
```
Fichero historial del bash
```
$HOME/.bash.history
```

ALIAS
Los alias se definen en `.bashrc`. Por ejemplo:

```
alias quiensoy='id'
alias usuarios='echo Usuarios;who'
```

OPERADORES LÓGICOS

`!=`	`-ne`	`==`	`-eq`
`>`	`-gt`	`<`	`-lt`
`>=`	`-ge`	`<=`	`-le`

LA ORDEN TEST

Verifica ciertos criterios, los criterios se aplican a archivos y comparación de números.

Opciones para archivos 0= Verdadero 1= Negativo		especial. Opciones para números	
`-f`	Si el archivo es ordinario		
`-d`	Si es un directorio	`-eq`	Igual
`-r`	Si es legible	`-ne`	Desigual
`-s`	Si existe y no está vacio	`-ge`	Mayor o igual que
`-w`	Si existe y tiene permiso de escritura	`-gt`	Mayor que
`-x`	Si existe y es ejectuable	`-le`	Menor o igual que
`-b`	Si es un archivo de bloque especial	`-lt`	Menor
`-c`	Si es un archivo de carácter		

REDES

DIRECCIONES DE RED

CLASE A
Desde 1.0.0.0 hasta 127.0.0.1. El 1er octeto define la red.
CLASE B
Desde 128.0.0.0 hasta 191.255.0.0. Los dos primeros octetos definen la red.
CLASE C
Desde 192.0.0.0 hasta 223.255.255.0. Los tres primeros octetos definen la red.
CLASES D,E y F
Las direcciones entre 224.0.0.0 hasta 254.0.0.0 son experimentales o están reservadas y no especifican ninguna red.

Direcciones especiales:
0.0.0.0 *Encaminamiento por defecto*
127.0.0.1 Reservada para *loopback*

Rangos de direcciones reservadas para redes locales:

CLASE	REDES
A	10.0.0.0 hasta 10.255.255.255
B	172.16.0.0 hasta 172.31.0.0
C	192.168.0.0 hasta 192.168.255.0

Máscaras habituales de red y correspondencia en bits

MASCARA	BITS
255.0.0.0	8
255.255.0.0	16
255.255.255.0	24
255.255.255.128	25
255.255.255.192	26
255.255.255.224	27
255.255.255.240	28
255.255.255.248	29
255.255.255.252	30

Dispositivos de red más comunes en linux:

lo	Interfaz de bucle local (loopback). Es usado fundamentalmente para realizar tests.
eth0,eth1...	Interfaces de los dispositivos ethernet. Son usados por la mayoría de las tarjetas Ethernet.
tr0,tr1...	Interfaces de los dispositivos Token Ring.
sl0,sl1...	Interfaces de dispositivos SLIP, se asocian a lineas seriel (por ejemplo, a un modem).
ppp0,ppp1...	Interfaces de dispositivos PPP, se asocian a lineas seriel (por ejemplo, a un modem).
plip0,plip1...	Interfaces PLIP, que transporta datagramas IP sobre lineas paralelas.
ax0,ax1...	Interfaces AX25. Es el principal protocolo usado por los radioaficionados.

FICHEROS RELACIONADOS

/etc/hosts
/etc/host.conf
/etc/resolv.conf
/etc/networks
/etc/nsswitch.conf

COMANDOS RELACIONADOS

hostname
ifconfig
arp
ping
route
setserial
netstat
nslookup

telnet
ssh
rlogin

ftp
talk

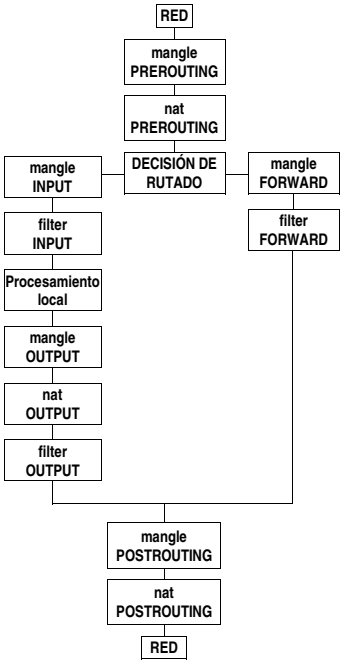
REDES - IPTABLES

Información extractada y traducida de: http://iptables-tutorial.frozenlinux.net/iptables-tutorial.html

¿EN QUÉ ORDEN CIRCULAN LOS PAQUETES A TRAVÉS DE LAS REGLAS DE IPTABLES?

Cuando un paquete entra en el cortafuegos, pasa de la capa hardware (tarjeta de red habitualmente) al driver apropiado del kernel. Entonces el paquete atraviesa una serie de pasos del kernel, que determinan que se entregue el paquete a la aplicación correcta (localmente), se reenvie a otro host (forward), o lo que corresponda.

ESQUEMA DE RUTADO DE LOS PAQUETES



Resumen de las tablas:

- **Tabla mangle**: No debe incluirse ningún tipo de filtrado en esta tabla, ni tampoco realizar DNAT, SNAT o Masquerading. Sólo son válidos en la tabla mangle los siguientes tarjets: TOS, TTL y MARK.

  TOS - Se usa para cambiar el *Type of Service.*
  TTL - Se usa para cambiar el *Time To Live.*
  MARK - Se usa para poner marcas especiales a un paquete.

- **Tabla nat**: Esta tabla sólo debería ser usada para realizar NAT en los diferentes paquetes. En otras palabras, solo debería de ser usado para traducir el campo de origen o el campo de destino del paquete. Anotar que sólo el primer paquete del datagrama alcanzará esta regla. Después de él, el resto de paquetes automáticamente tomará la misma acción que ha tomado el primer paquete. Los actuales tarjets aplicables a la tabla nat los los siguientes: DNAT, SNAT y MARQUERADE.

  DNAT – Traducción de Direcciones de Red de Destino. Se usa mayormente cuando por una direccion IP pública (internet) entra un paquete en el cortafuegos y se redirecciona a otro host de la red (un servidor web detrás del cortafuegos, por ejemplo).
  SNAT – Traducción de Direcciones de Red de Origen. Se usa mayormente para cambiar la dirección origen de un paquete. Un buen ejemplo es conectar una red local a internet con sólo una IP pública.
  MARQUERADE – Este tarjet de usa de la misma manera que SNAT, pero MASQUERADE exige más trabajo por parte del cortafuegos. La razón de esto es que cada vez que un paquete alcanza el tarjet MASQUERADE, automáticamente comprueba la IP a usar, al contrario que SNAT, que simplemente utiliza la IP que hemos definido. El tarjet MASQUERADE hace posible que podamos trabajar correctamente con las direcciones dinámicas DHCP que muchos ISP ofrecen con sus conexiones a internet.

- **Tabla filter**: La tabla filter se usa mayormente para filtrar paquetes. Podemos coger paquetes y filtrarlos de la manera que queramos. Desde luego, podemos hacer un filtrado previo; aunque de todos modos esta tabla en particular es el lugar diseñado específicamente para realizar el filtrado.

RUTADO DE UN PAQUETE DESTINADO A OTRO ORDENADOR

Paso	Tabla	Regla	Comentario
01			En el cable (p.ej. Internet)
02			Entra en la interfaz (p.ej. eth0)
03	mangle	PREROUTING	Esta regla se usa normalmente para el manejo de paquetes (p.ej. cambiar TOS – Type of
04	nat	PREROUTING	Esta regla se usa para hacer Traducción de Direcciones de Red de Destino (DNAT). La Traducción de Direcciones de Red de Origen (SNAT) se realiza más adelante. Se recomienda no hacer ningún tipo de filtrado en esta regla ya que podría ser traspasada en ciertos casos.
05			Decisión de rutado (p.ej. si el paquete se destina a este mismo ordenador o si se reenvía, y a donde).
06	mangle	FORWARD	El paquete es entonces enviado a la regla FORWARD de la tabla mangle. Esto puede ser usado para necesidades muy específicas, en donde queremos manejar los paquetes después de la decisión inicial de rutado, pero antes de la última decisión de rutado realizada justo antes de que el paquete fuera enviado.
07	filter	FORWARD	El paquete es rutado dentro de la regla FORWARD. Sólo los paquetes reenviados pasarán por aquí, y aquí es donde se hará el filtrado. Resaltar que todo el tráfico reenviado transcurre por aquí (no solo en una dirección), y habrá que tomarlo en cuenta cuando escribamos nuestras reglas.
08	mangle	POSTROUTING	Esta regla se usa para manejar ciertos tipos de paquetes específicos que queremos se sitúen después de que se hayan realizado todos los tipos de decisión de rutado, pero dentro de esta máquina.
09	nat	POSTROUTING	Esta regla debería de ser la primera que se utilice para hacer SNAT. Se recomienda no hacer ningún tipo de filtrado en esta regla ya que podría ser traspasada en ciertos casos. También es donde se hace el MASQUERADING.
10			Sale de la interfaz (p.ej. eth1)
11			Sale por el cable de nuevo (p.ej. Internet)

RUTADO DE UN PAQUETE DESTINADO A ESTE ORDENADOR

Paso	Tabla	Regla	Comentario
01			En el cable (p.ej. Internet)
02			Entra en la interfaz (p.ej. eth0)
03	mangle	PREROUTING	Esta regla se usa normalmente para el manejo de paquetes (p.ej. cambiar TOS – Type of Service)
04	nat	PREROUTING	Esta regla se usa para hacer Traducción de Direcciones de Red de Destino (DNAT) principalmente.
05			Decisión de rutado (p.ej. si el paquete se destina a este mismo ordenador o si se reenvía, y a donde).
06	mangle	INPUT	En este punto, se alcanza la regla INPUT. Se usa esta regla para manejar paquetes después de que hallan sido rutados, pero antes de que salgan para su procesado por el ordenador.
07	filter	INPUT	Aquí es donde se realiza el filtrado de todo el tráfico de entrada destinado a nuestro host local. Anotar que todos los paquetes destinados al host local pasarán por esta regla, no importa de qué interfaz o de qué dirección provenga.
08			Proceso local/aplicación (p.ej. programa cliente/ servidor)

RUTADO DE UN PAQUETE QUE SALE DE ESTE ORDENADOR

Paso	Tabla	Regla	Comentario
01			Proceso local/aplicación (p.ej. programa cliente/ servidor)
02			Decisión de rutado. Qué dirección de origen se usa, qué interface de salida, y otra información necesaria que necesita ser recogida.
03	mangle	OUTPUT	Esta regla se usa normalmente para el manejo de paquetes. Se recomienda no hacer ningún tipo de filtrado en esta regla ya que puede tener efectos no deseados.
04	nat	OUTPUT	Esta regla se usa para hacer Traducción de Direcciones de Red (NAT) de los paquetes que salen desde el cortafuegos (host local).
05	filter	OUTPUT	Aquí es donde se filtran los mensajes que salen del host local.
06	mangle	POSTROUTING	La regla POSTROUTING de la tabla mangle se usa principalmente cuando queremos manejar paquetes antes de que abandonen el host, pero después de las decisiones de rutado. Alcanzarán esta regla tanto los paquetes que atraviesen el cortafuegos como los paquetes creados por él mismo.
07	nat	POSTROUTING	Aquí es donde se realiza SNAT. Se recomienda no hacer ningún tipo de filtrado en esta regla ya que puede tener efectos no deseados.
08			Sale de la interfaz (p.ej. eth1)
09			Sale por el cable de nuevo (p.ej. Internet)

COMO SE CREA UNA REGLA

```
iptables -t tabla comando \
  [coincidencia] [objetivo]
```

`-t tabla`
Si no se especifica tabla, se utiliza filter por defecto.

`comando`
Siempre debe de ir primero, o inmediatamente después de la especificación de la tabla. Usamos *comando* para decir al programa qué hacer, por ejemplo insertar una regla o borrarla.

`coincidencia`
Coincidencia (*match* en inglés) es la parte de la regla que especifica el carácter del paquete, que lo hace diferente de otros paquetes. Aquí podemos especificar desde qué IP viene el paquete, que interface de red, el puerto, el protocolo o lo que sea.

`objetivo`
Finalmente tenemos el objetivo (o *tarjet*) del paquete. Si se da la coincidencia (*match*), decimos al kernel qué hacer con él. Podemos por ejemplo, decir al kernel que mande el paquete a otra regla que hayamos creado nosotros mismos, la cual es parte de esta tabla en particular. También podemos decirle al kernel que descarte ciertos paquetes, o que los devuelva al remitente.

Comandos de Iptables

-A —append Inserta una regla al final de una tabla
-D —delete Borra una regla
-R —replace Reemplaza una regla por otra
-I —insert # Inserta una regla en algún lugar determinado de una tabla
-L —list Muestra todas las reglas de una tabla
-F —flush Borra todas las reglas de una tabla
-Z —zero Pone a 0 todos los contadores de una tabla determinada.
-N —new-chain Crea una subtabla dentro de una tabla especificada.
-X —delete-chain Borra una tabla de usuario (es necesario que no contenga reglas en ella). Si se usa sin ninguna opción, borra todas las tablas de usuario vacias.
-P —policy Le dice al kernel que establezca una política por defecto. Todos los paquetes que no coincidan con alguna regla estarán forzados a usar la política por defecto (DROP, ACCEPT o REJET).
-E —rename-chain Cambia el nombre de una tabla de usuario por otra.

Ejemplos

```
iptables -A input -p tcp -i\
eth0 —dport 80 -j drop
```
Cierra las conexiones entrantes desde eth0 con destino al puerto local 80 (http).

```
iptables -t nat -A prerouting \
-i eth1 -p tcp —dport 80 -j \
REDIRECT —to-port 3128
```
Redirecciona al puerto 3128 (proxy) todos los paquetes que entran por eth1 y tienen por destino el puerto 80 (http).

Sugerencias para el uso de máscaras de bits de TOS			
TOS	**ANDmask**	**XORmask**	**Uso suregido**
Demora mínima	0x01	0x10	ftp,telnet,ssh
Rendimiento máximo	0x01	0x08	ftp-data,www
Fiabilidad máxima	0x01	0x04	snmp,dns
Coste mínimo	0x01	0x02	nntp,smtp