
Configuración de la red

Diseño y Administración de Sistemas y Redes

Juan Céspedes <cespedes@gsync.es>



Curso 2005–2006



Configuración de la red

- De manera permanente: modificando una serie de ficheros de configuración para que la red se configure en el arranque de la máquina.

No está estandarizada la localización y el formato de muchos de los ficheros de configuración de la red; cada distribución de utiliza ficheros y directorios distintos. Por ejemplo, Red Hat utiliza `/etc/sysconfig/network*`, mientras que Debian utiliza `/etc/network/*`.

- De manera inmediata: utilizando órdenes que cambian en el acto los parámetros de configuración, como con `ifconfig`, `route` o `ip`.

Configuración permanente

- Nombre de la máquina:
 - `/etc/HOSTNAME` (Red Hat)
 - `/etc/hostname` (Debian)
- Configuración de cada interfaz de red:
 - `/etc/sysconfig/network-scripts` (Red Hat)
 - `/etc/network/interfaces` (Debian)
- Configuración adicional:
 - `/etc/sysconfig/network` (Red Hat)
 - `/etc/network/options` (Debian)

Configuración específica de Debian

- `/etc/network/interfaces:`

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
    address 192.168.1.5
    netmask 255.255.255.0
    gateway 192.168.1.1

iface eth1 inet dhcp
```

- `/etc/network/options:`

```
ip_forward=no
spoofprotect=yes
syncookies=no
```

ifup / ifdown

- Utilizados por las distribuciones para activar y desactivar interfaces de red.
- Configuran el interfaz especificado (o todos los automáticos si se especifica la opción “-a”), según lo indicado en los ficheros de configuración.
- Es la manera preferida de activar y desactivar un interfaz dentro de una distribución.

ifconfig

- Herramienta de *bajo nivel* para mostrar información acerca de la configuración de uno o varios interfaces de red, o para configurar uno de ellos.
- Uso:

```
ifconfig interface [ opciones ]
```

Posibles opciones:

- **up**, **down**: Activa / desactiva el interfaz.
- **dirección IP**: Asigna esa dirección IP al interfaz.
- **netmask máscara**: Asigna esa dirección como *netmask*.
- **[-]broadcast [dirección]**: Si se proporciona dirección, asigna la dirección de *broadcast*; si no, activa o desactiva la opción de *broadcast* en el interfaz.

ifconfig (2)

- Otras opciones de `ifconfig`:
 - `[-]arp`: Activa o desactiva el uso del protocolo ARP en el interfaz.
 - `[-]promisc`: Activa o desactiva el modo promiscuo del interfaz.
 - `[-]pointopoint [dirección]`: Activa el modo “punto a punto” de un interfaz, y especifica la dirección IP del otro extremo.
 - `hw ether dirección`: Modifica la dirección MAC, o dirección *hardware* del interfaz.

route

- Muestra la tabla de rutas, o la modifica añadiendo o eliminando una determinada ruta.
- Uso:

```
route [-n] [add|del [-net|-host] dirección...  
...[netmask mask] [gw GW] [dev if]]
```
- La opción “add” o “del” llevan a cabo el añadir o borrar una regla de la tabla de rutas.
- Si se usa “-net” es obligatorio especificar “netmask”.
- “-host dirección” equivale a “-net dirección netmask 255.255.255.255”.
- No usar “-host” ni “-net” equivale a usar “-host”.
- “default” equivale a “-net 0.0.0.0 netmask 0.0.0.0”.

Ejemplos

- `ifconfig eth0 192.168.1.5 netmask 255.255.255.0`
- `route add default gw 192.168.1.1`
- `route add -net 192.168.2.0 netmask 255.255.255.0 dev eth0`
- `route add -net 192.168.5.0 netmask 255.255.255.0 gw 192.168.1.5`
- `route add -host 1.2.3.4 gw 192.168.1.4`
- `route add -net 2.3.4.5 netmask 255.255.255.255 dev eth0`
- `route del -host 2.3.4.5`

Configuración automática de la red

- En muchas ocasiones, no conocemos la dirección IP que debemos usar para configurar la red, sino que nuestro proveedor nos la proporciona automáticamente.
- Métodos de configuración automática:
 - DHCP: Es el más habitual en conexiones *ethernet*. Para configurar la red en Linux, hay que usar alguna de las órdenes `dhclient`, `pump`, `udhcpc` o `dhcpcd`.
 - BOOTP: Fue el primero, pero ya casi no se utiliza. Existen clientes específicos de BOOTP, como `bootpc`, y algunos clientes de DHCP pueden usar también este protocolo, como `pump`.
 - PPP: El más utilizado en conexiones telefónicas. Gestionado automáticamente por el servidor de PPP, `pppd`.

Servidores de red

- Cada servidor de red utiliza un protocolo y un servicio determinado para atender peticiones.

- Lista de protocolos en “/etc/protocols”:

```
icmp    1  ICMP # internet control message protocol
tcp     6  TCP  # transmission control protocol
udp    17  UDP  # user datagram protocol
...
```

- La lista de servicios asociados a cada protocolo (TCP o UDP) está en el fichero “/etc/services”:

```
ftp     21/tcp
ssh     22/tcp
telnet  23/tcp
smtp    25/tcp
domain  53/udp
...
```

inetd

- Existen determinados servicios que suelen estar siempre activos, aunque raramente se utilizan.
- Para evitar que haya muchos procesos funcionando, y que cada uno de ellos tenga que hacer básicamente lo mismo (establecer conexión TCP o UDP, utilizar un determinado puerto, atender peticiones, enviar resultados), existe un “superdemonio” que se encarga de todas estas tareas y ejecutar otro proceso cuando se recibe una conexión: **inetd**.
- **inetd** lee un fichero de configuración, el “/etc/inetd.conf”, donde se indica en qué puertos escuchar y qué órdenes ejecutar con cada conexión.

inetd.conf

- Cada línea indica un servicio nuevo, en el que tiene que escuchar y ejecutar un proceso cuando le lleguen conexiones.
- Sintaxis:

```
servicio tipo protocolo wait/nowait[.max]...  
...user[.group] programa argumentos
```
- **servicio, protocolo**: Tiene que coincidir con alguno especificado en el fichero `/etc/services`.
- **tipo**: Ha de ser “`stream`” para TCP y “`dgram`” para UDP.
- **wait/nowait**: Solo es aplicable para el protocolo UDP, indica si un proceso gestiona varios paquetes (“`wait`”) o si `inetd` debe seguir recibiendo y creando nuevos procesos (“`nowait`”).

inetd.conf (2)

- **max**: Número que indica el número máximo de procesos de un tipo que pueden estar lanzados simultáneamente.
- **user, group**: Identidad con que se ejecutará el proceso, o “`internal`” para servicios gestionados internamente por `inetd`.
- **programa**: Nombre del proceso a ejecutar.
- **argumentos**: Lista de argumentos, comenzando con `argv[0]`, esto es, con el *nombre del proceso* que se verá al ejecutarlo.

Ejemplo:

```
echo  stream tcp nowait root      internal  
telnet stream tcp nowait telnetd  /usr/sbin/tcpd  /usr/sbin/in.telnetd  
ident stream tcp wait  identd   /usr/sbin/identd  identd  
talk  dgram  udp  wait  nobody.tty /usr/sbin/in.talkd  in.talkd
```

Configuración avanzada de la red: *iproute*

- En los últimos años se han hecho muchos avances en las posibilidades de configuración de la red y las rutas en el *kernel* de Linux.
- Las nuevas características se pueden controlar mediante el uso de un programa conocido como “*iproute*”, pero cuyo ejecutable en realidad se llama “**ip**”.

```
# ip
Usage: ip [ OPTIONS ] OBJECT { COMMAND | help }
where OBJECT := { link | addr | route | rule | neigh | tunnel |
                maddr | mroute | monitor | xfrm }
        OPTIONS := { -V[ersion] | -s[tatistics] | -r[esolve] |
                    -f[amily] { inet | inet6 | ipx | dnet | link } |
                    -o[neline] }
```

iproute (2)

- **iproute** se comporta de distintas maneras, dependiendo de cuál sea el “objeto” a mostrar o modificar. Los “objetos” pueden ser:
 - **link**: Configuración *hardware* de un interfaz.
 - **addr**: Direcciones IP asignadas a un interfaz de red.
 - **route**: Tabla de rutas.
 - **rule**: Reglas de búsqueda de tablas de rutas.
 - **neigh**: Tabla ARP.
 - **tunnel**: Gestión de túneles IP.

ip link

- Muestra o modifica parámetros de la configuración de un interfaz de red (si está activo o no, dirección *hardware*, opciones del interfaz, etc):

```
# ip link
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 1000
   link/ether 08:00:46:f3:f9:22 brd ff:ff:ff:ff:ff:ff
4: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop qlen 1000
   link/ieee1394 08:00:46:03:01:c2:9f:08 brd ff:ff:ff:ff:ff:ff:ff:ff
5: sit0: <NOARP> mtu 1480 qdisc noop
   link/sit 0.0.0.0 brd 0.0.0.0
```

- “**ip -s link**” muestra estadísticas (bytes y paquetes enviados y recibidos, errores, etc).
- “**ip link help**” muestra todas las opciones para modificar parámetros de interfaces.

ip addr

- Muestra prácticamente la misma información que “**ip link**”, y además, las direcciones IP asociadas a cada interfaz.

```
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
   inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 1000
   link/ether 08:00:46:f3:f9:22 brd ff:ff:ff:ff:ff:ff
   inet 192.168.64.10/24 brd 192.168.62.255 scope global eth0
   inet6 fe80::a00:46ff:fef3:f922/64 scope link
       valid_lft forever preferred_lft forever
4: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop qlen 1000
   link/ieee1394 08:00:46:03:01:c2:9f:08 brd ff:ff:ff:ff:ff:ff:ff:ff
5: sit0: <NOARP> mtu 1480 qdisc noop
   link/sit 0.0.0.0 brd 0.0.0.0
```

- Se muestra una línea por cada dirección IP, pero, a diferencia de **ifconfig**, puede haber varias direcciones IP diferentes asociadas a un solo interfaz.

ip addr (2)

```
Usage: ip addr {add|del} IFADDR dev STRING
       ip addr {show|flush} [ dev STRING ]
```

- IFADDR tiene la forma “**ip/mask**”, donde “**mask**” es el número de bits a 1 en la *netmask*.
- Al “añadir” una dirección IP *no se borran* las direcciones IP que hubiera antes; la nueva se añade a la lista de las que ya tuviera.
- “**ip addr flush**” se utiliza para borrar todas las direcciones IP asociadas a un interfaz.

ip route

- Muestra una tabla de rutas; por defecto, la tabla de rutas “principal” del *kernel*, la misma que muestra “**route**”:

```
192.168.64.0/24 dev eth0 proto kernel scope link src 192.168.64.10
default via 192.168.64.1 dev eth0
```

- En el *kernel* puede haber varias tablas de rutas distintas (y de hecho las hay; lo veremos más adelante). Para mostrar una tabla de rutas distinta, hay que usar “**ip route list table *tabla***”. Por ejemplo:

```
# ip route list table local
broadcast 192.168.64.255 dev eth0 proto kernel scope link src 192.168.64.10
broadcast 127.255.255.255 dev lo proto kernel scope link src 127.0.0.1
local 192.168.64.10 dev eth0 proto kernel scope host src 192.168.64.10
broadcast 192.168.64.0 dev eth0 proto kernel scope link src 192.168.64.10
broadcast 192.168.62.255 dev eth0 proto kernel scope link src 192.168.64.10
broadcast 127.0.0.0 dev lo proto kernel scope link src 127.0.0.1
local 127.0.0.1 dev lo proto kernel scope host src 127.0.0.1
local 127.0.0.0/8 dev lo proto kernel scope host src 127.0.0.1
```

ip route (2)

```
Usage: ip route { list | flush } [ table TABLE-ID ]
       ip route get ADDRESS
       ip route { add | del | change | replace } ROUTE
               [ via ADDRESS ] [ dev STRING ]
```

- “**list**”: Muestra la lista de rutas de una tabla dada.
- “**flush**”: Borra la lista de rutas.
- “**get**”: Indica la ruta que seguiría un paquete hacia esa dirección.
- “**add, del, change, replace**”: Modifica la tabla de rutas, añadiendo, borrando o cambiando una ruta.
- “**ROUTE**” tiene que expresarse como “**ip/mask**”, igual que las direcciones IP en “**ip addr**”.

ip rule

- Muestra la lista de tablas de rutas del sistema, indicando el orden en el que se consultan y qué tipo de rutas se consultan en cada tabla:

```
0:      from all lookup local
32766:  from all lookup main
32767:  from all lookup default
```
- “**ip rule**” permite conseguir rutado en función de parámetros que normalmente no se tienen en cuenta, como el interfaz de llegada, la dirección IP de origen, el campo “TOS” del paquete IP, etc.
- Normalmente no se usa ni se modifica, a no ser que tengamos requisitos muy especiales de rutado en nuestra red.

ip neigh

- Se comporta de una manera muy similar a la orden “arp”, permitiendo consultar y cambiar la tabla ARP del *kernel*:

```
192.168.64.1 dev eth0 lladdr 00:0e:0c:06:7d:b4 nud stale
```

- Al igual que “arp”, permite modificar esta tabla, añadiendo, borrando o modificando entradas, pero tampoco suele usarse a no ser que queramos hacer cosas extrañas con la red, como permitir que nuestra máquina conteste a peticiones ARP que no van destinadas a ninguna dirección IP propia.

ip tunnel

- Permite establecer túneles usando IPv4 o IPv6, permitiendo que estamos utilizando unas direcciones IP que en realidad no están en nuestra red, sino en otra lejana, encapsulando todos los paquetes y enviándolos a través del túnel hacia el destino (y al revés, recibiendo los paquetes del destino para poder verlos desde nuestra máquina).

- Uso:

```
Usage: ip tunnel { add | change | del | show } [ NAME ]  
        [ mode { ipip | gre | sit } ]  
        [ remote ADDR ] [ local ADDR ]
```

Como puede verse, para cada túnel podemos usar un modo distinto, que indica el tipo de encapsulado, junto con las direcciones IP local y remota (de cada extremo del túnel).