Uso de la red en Linux

Objetivo

Familiarizarse con el uso de aplicaciones y servicios de red en Linux.

Cambiando el PATH

En la siguiente sección utilizará el comando `ifconfig`, y la shell probablemente le dirá que no puede encontrarlo. Esto es así porque la shell no busca comandos en todos los directorios, sino sólo en los que tiene configurados para ello, y su ordenador no está configurado para que busque en el directorio /sbin que es donde está dicho comando. Si quiere, puede modificar los directorios en los que la shell busca comandos y aprender a configurarlo correctamente, tal y como se detalla a continuación, o bien, utilizar directamente el camino absoluto del comando: /sbin/ifconfig.

Hasta ahora cuando ha utilizado un comando, la shell se ha encargado de encontrar el programa adecuado en el disco y lanzar el programa correspondiente. Esto se debe a que los programas que hemos usado son muy comunes y estaban almacenados en los directorios en que típicamente se guardan los comandos de UNIX (`/bin`,`/usr/bin`)

Puede probar el comando `which` que le indica dónde ha encontrado la shell un comando concreto.

```
$ which date
$ which cat
```

Podemos indicar a la shell la localización del programa que queremos ejecutar indicando el camino al comando en lugar del comando

```
$ /bin/cat /etc/services
  tiene el mismo efecto que
$ cat /etc/services
  con la unica diferencia que la shell no busca el comando en el disco duro
  sino que va directamente a /bin a buscarlo
```

Pero aún en el caso en que la shell deba buscar el fichero del comando, no hace una búsqueda por todo el disco sino que simplemente mira en una lista de directorios en los que comúnmente están los comandos. Esta lista se almacena en una variable de entorno del shell que se denomina PATH. Puede ver el valor de esa variable con el siguiente comando. (La sintaxis para acceder a las variables en la shell es poner un símbolo $ delante del nombre de la variable que será sustituida por su valor)

```
$ echo $PATH
/usr/local/bin:/bin:/usr/bin
```

Observe que el contenido de la variable es una serie de directorios separados por :. La shell busca comandos sólo en esos directorios y lo hace en el orden en que aparecen en la variable. Modificando la variable conseguirá que busque en otros directorios. Por ejemplo es probable que en las prácticas con la red, quiera acceder a comandos que estén en el directorio `/sbin`. Puede añadir el directorio `/sbin` a la variable `PATH` como se indica a continuación (tenga en cuenta que las variables se manejan de forma diferente en las diferentes tipos de shell, las instrucciones que siguen sólo son aplicables a las shells de tipo `bash`)

```
    Asi se da valor a una variable de la shell
$ A=1
    veamos si funciona
$ echo "A=$A"
    Asi se da valor a una variable que sera heredada por los procesos hijos
$ export A=1
    asi podemos modificar el PATH
$ export PATH=$PATH:/sbin
$ echo "PATH=$PATH"
```

Compruebe que una nueva shell lanzada desde ésta recibe la variable `PATH` modificada

Compruebe también que la variable sólo se propaga a los procesos descendientes de la shell en la que la hemos cambiado. Si, por ejemplo, abre un terminal nuevo, no tendrá la variable.

Para modificar la variable PATH de forma que afecte a cualquier nueva shell que abra deberá modificar el fichero de arranque de la shell. Cuando se lanza `bash` lee una serie de ficheros entre los que se encuentra el fichero `.bashrc` en su directorio *home* y ejecuta los comandos de esos ficheros. El fichero `.bashrc` es el fichero apropiado para poner modificaciones a la variable `PATH`. Puede editarlo y colocar ahí la línea que modifique la variable. Compruebe que todas las shells que abre ahora tienen la variable modificada.

¿Qué necesito para estar en red?

Un ordenador que vaya a funcionar en red necesita una dirección para que la red pueda dirigir hacia él los datos que le envían el resto de ordenadores. Como hemos visto esta es la dirección IP. De momento debemos saber que un ordenador para estar conectado a Internet necesita una dirección IP. Para ver la dirección IP de su ordenador puede usar el comando ifconfig.

```
$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:11:2F:42:0C:1B
          inet addr:10.1.1.11  Bcast:10.1.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:463154 errors:0 dropped:0 overruns:0 frame:0
          TX packets:238027 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:69571693 (66.3 Mb)  TX bytes:40092460 (38.2 Mb)
          Interrupt:10 Base address:0xd800

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:23546 errors:0 dropped:0 overruns:0 frame:0
          TX packets:23546 errors:0 dropped:0 overruns:0 carrier:0
```

```
          collisions:0 txqueuelen:0
          RX bytes:3350286 (3.1 Mb)  TX bytes:3350286 (3.1 Mb)
```

El comando `ifconfig` le muestra información sobre los diferentes interfaces del ordenador conectados a la red. A lo largo del curso veremos varios tipos de interfaces posibles. De momento puede observar que su ordenador tiene dos interfaces:

- `eth0` es una conexión a una red de área local Ethernet y es la verdadera conexión de red de ese ordenador.

- `lo` es un interfaz ficticio llamado interfaz de loopback todo lo que se envía por ese interfaz se vuelve a recibir en el ordenador. Es típico de los sistemas UNIX tener este interfaz y vale para enviarse datos a si mismo incluso cuando el ordenador no esta conectado a la red. En los sistemas UNIX muchas partes del sistema operativo funcionan como servicios de red, de ahí que el interfaz de loopback sea muy útil. Pero de momento no se preocupe por él.

Así pues su ordenador tiene un interfaz conectado a una red Ethernet y en dicho interfaz utiliza la dirección IP que se ve en el campo `inet addr:`. Compruebe cuál es su dirección IP. Esa dirección es suficiente para identificar a su ordenador en Internet. En nuestro caso, al estar la red del Laboratorio separada de Internet (es una Intranet) la dirección sólo le identifica entre los ordenadores del dominio del Área de Telemática (o sea este laboratorio[Telemática 2] y el de arriba[Telemática 1]). Pero para todos los efectos funciona igual que Internet.

Pruebe el comando `ping`. El comando `ping` es una utilidad que le permite comprobar si existe conectividad de red entre dos máquinas. Para ello la máquina que lanza el comando `ping` envía paquetes del protocolo ICMP que el sistema operativo de la máquina de destino está obligada a responder al origen. El comando `ping` recibe estos paquetes y nos los muestra indicándonos también el tiempo que tardan en ir y volver y contando los que se pierden. Mire la dirección IP que tiene su vecino de mesa y haga ping a su ordenador y al del vecino.

```
$ ping direccion_IP_de_mi_vecino
$ ping mi_direccion_IP
```

Observe la diferencia de tiempos. Pruebe con otros vecinos. ¿Cómo hace ping para saber que los paquetes se pierden?

Por otra parte probablemente habrá observado que en Internet los usuarios no manejan direcciones IP para referirse a otras máquinas. Por ejemplo, no reconocerá que la dirección IP 64.233.183.99 pertenece al buscador más popular, sino que lo recordará más bien con el nombre de www.google.com. Esto es así porque existe un servicio que nos permite dar nombres a las direcciones IP para recordarlas más fácilmente. Este sistema se llama *Sistema de Nombres de Dominio* o *DNS*. A pesar de su utilidad este sistema es simplemente un servicio sobre Internet. Cuando nosotros usamos un nombre como www.google.com, nuestro ordenador envía mensajes a ciertos servidores a los que conoce por su dirección IP para preguntarles cuál es la dirección IP asociada a www.google.com. Estos ordenadores la buscan en sus tablas de direcciones y se la envían. Seguidamente nuestro ordenador utiliza la dirección IP correspondiente para comunicarse con el buscador.

Es decir que desde el punto de vista de la red todo se hace con direcciones IP, los nombres son un simple añadido que se traduce a dirección IP a la menor oportunidad.

En el laboratorio de telemática está disponible el servicio de nombres. Puede probarlo con los comandos `host` o `nslookup` que son prácticamente equivalentes.

```
$ host tlm11.net.tlm.unavarra.es
    búscame la direccion IP de tlm11.net.tlm.unavarra.es
$ host tlm11
    búscame la direccion IP de tlm11 en el dominio en el que estoy
$ host 10.1.1.11
    búscame el nombre para esta dirección IP

    o bien
$ nslookup tlm11.net.tlm.unavarra.es
$ nslookup tlm11
$ nslookup 10.1.1.11
```

En general, todos los comandos que utilizan direcciones de red aceptan nombres o direcciones IP por igual. Pruebe por ejemplo el ping utilizando nombres.

También puede averiguar el nombre de su ordenador utilizando el comando `hostname`.

Ahora que ya sabemos que es estar en la red y como se llaman otros ordenadores, probemos algún servicio.

Acceso remoto y transferencia de ficheros

¿Cuáles cree que fueron las primeras aplicaciones de la red? Antes del correo electrónico, mucho antes del web, lo realmente interesante de tener ordenadores en red era poder acceder remotamente a otros ordenadores (recordemos que estábamos en un mundo de grandes ordenadores centrales y muchos usuarios) y por supuesto poder transferir ficheros entre ordenadores.

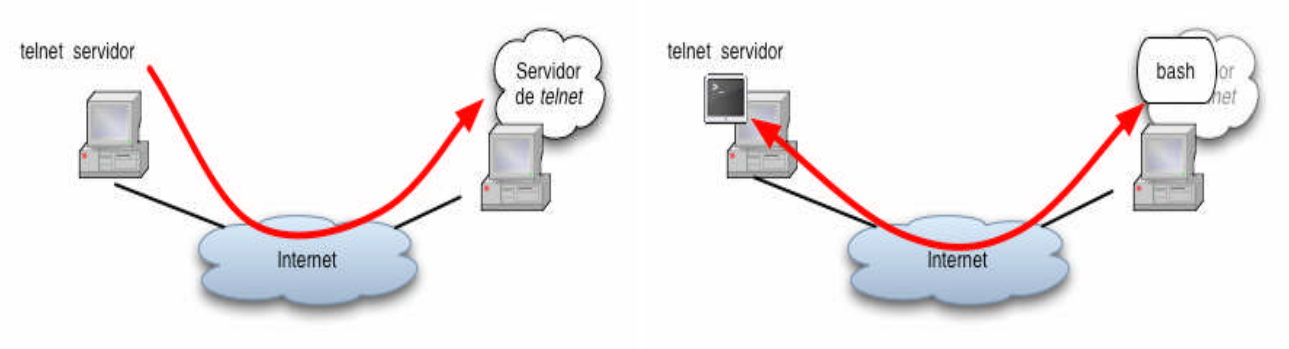
El servicio de acceso remoto por excelencia se llama `telnet`. Para utilizarlo debe usar el comando `telnet` y decirle qué ordenador debe de usar de forma remota. Por ejemplo intente conectarse para acceder remotamente al ordenador jupiter. Todos estos comandos deberían funcionar.

```
$ telnet jupiter.net.tlm.unavarra.es
$ telnet jupiter
$ telnet 10.1.1.230
```

Compruebe que puede *hacer login* en dicho ordenador *con su cuenta*. ¿Cómo puede comprobar que está en otro ordenador? Observe la dirección IP del ordenador remoto.

¿Cómo funciona `telnet`? Al hacer telnet a un ordenador remoto su ordenador envía paquetes para establecer una conexión con el ordenador remoto. Pero debe existir un programa en el ordenador remoto que esté esperando recibir conexiones de telnet. Una vez establecida la conexión el programa *servidor de telnet* recibe los comandos del programa *cliente de telnet* los ejecuta en la máquina remota y nos devuelve los resultados a través de la red. Básicamente el *servidor de telnet*

lanza una shell y redirecciona la salida y la entrada de la shell a través de la red al *cliente de telnet* en nuestro ordenador.



La transferencia de ficheros clásica tiene un funcionamiento parecido. En este caso la aplicación clásica se llama FTP (File Transfer Protocol). En su ordenador tiene el comando `ftp` y puede probarlo también transfiriendo ficheros a `jupiter`.

Pero tenga cuidado dado que su directorio home está compartido por la red en todos los ordenadores del laboratorio, no tiene sentido transferir ficheros que estén en su home pruebe a transferir ficheros a un directorio que no este montado por la red.

Por ejemplo vaya al directorio `/tmp` y use los siguientes comandos para transferir un fichero desde `jupiter`

```
$ cd /tmp
$ ftp jupiter
```

El ordenador destino le pedirá que introduzca un usuario y contraseña y tras introducirlo podrá moverse por los directorios del destino con `cd` y `ls`. Puede usar también el comando `get` para coger un fichero y traerlo al odenador local y `put` para subir un fichero desde el ordenador local.

Al igual que sucede con `telnet` es necesario que exista un *servidor de ftp* escuchando en el ordenador remoto para que podamos utilizar el servicio. En `jupiter` estos servicios están activados pero puede comprobar que no responden en el resto de ordenadores del laboratorio.

Esto es así porque los servicios `telnet` y `ftp` tienen problemas de seguridad. Fueron diseñados en una época en la que las redes eran un recurso de investigación y en general los usuarios eran de confianza. Pero al extenderse Internet al trafico comercial y permitir el acceso sin restricciones ya no ofrecen garantías de seguridad. Básicamente `telnet` y `ftp` no protegen los datos ante posibles usuarios que intercepten los paquetes por lo que es relativamente fácil obtener las contraseñas que se usan. Así que actualmente es más normal utilizar herramientas de acceso remoto y transferencia de ficheros protegidas por técnicas criptográficas. Los equivalentes a `telnet` y `ftp` con seguridad se llaman `ssh` (secure shell) y `sftp` (secure ftp) o `scp` (secure copy) que son en realidad tres aplicaciones del mismo protocolo. El servicio de `ssh` sí que estará activado en todos los ordenadores del laboratorio y puede probarlo con el ordenador de su vecino.

Para conectarse con acceso remoto a un ordenador mediante `ssh` debe de usar el comando:

```
$ ssh cuenta@destino
```

Que permite acceder remotamente a la máquina destino utilizando el usuario cuenta. Si no especifica el usuario:

```
$ ssh destino
```

Se conectará al ordenador destino utilizando el mismo usuario que tiene en la máquina local.

Observe que la primera vez que se conecta a un ordenador con `ssh` el comando le informará de que no puede establecer la identidad del destino y le mostrará una serie de números, lo que se conoce como *fingerprint* (literalmente huella dactilar). El motivo de ésto es que las técnicas criptográficas no pueden resolverlo todo y aunque pueden establecer un canal seguro con un ordenador al que nunca se han conectado, no hay ninguna manera de probar que el ordenador destino es quien dice ser.

Por tanto la única forma de verificar la identidad sería consultar con el administrador de la máquina y comprobar que el *fingerprint* es correcto. Es decir el comando `ssh` nos está garantizando que hemos establecido un canal seguro con el ordenador cuya clave privada tiene ese *fingerprint*, pero es responsabilidad nuestra verificar que ese *fingerprint* corresponde al ordenador que queremos ir. Normalmente, a no ser que nos dediquemos al espionaje internacional, podemos decir que sí, sin mirar. Al hacer esto si nadie ha interceptado la conexión inicial, las siguientes veces que nos conectemos podemos estar seguros de que estamos conectándonos al mismo ordenador que la primera vez, ya que `ssh` se acordará del *fingerprint* de cada ordenador que ha visitado y nos avisará si cambia.

Pruebe a conectarse de forma segura al ordenador de al lado. Una vez allí, haga `ifconfig` y `hostname` para comprobar la identidad de la máquina en la que está.

Puede observar quién más está usando una máquina utilizando el comando `who` o `w`. Obtendrá un listado de usuarios parecido a este:

```
tlm111     pts/1        Oct 12 18:37 (:0.0)
tlm164     pts/2        Oct 12 18:39 (:0.0)
tlm176     pts/3        Oct 12 18:40 (:0.0)
```

Observe que para usuarios remotos `who` nos dice además desde qué dirección se están conectando. El comando `w` es parecido pero nos muestra además cuál es el comando que está ejecutando cada usuario o cuánto tiempo lleva sin hacer nada.

Observe que un usuario puede estar usando el sistema varias veces. Esto puede ser porque se haya conectado desde varios sitios, o bien por que tenga varias ventanas abiertas con terminales. En cualquier caso cada usuario está identificado también por un nombre de terminal en ese caso pts/1, pts/2 ...

Como curiosidad, una vez que tienes múltiples usuarios utilizando la misma máquina, enseguida a alguien se le ocurre que sería buena idea poder enviar y recibir mensajes entre los

usuarios. Desde muy antiguo los sistemas UNIX permitían dejar mensajes de unos usuarios a otros mediante un comando `mail`. De hecho este es el origen del correo electrónico de Internet. No vamos a probar el mail en las prácticas ya que normalmente los sistemas UNIX actuales tienen deshabilitado el mail por seguridad. Pero no es el mail la única forma de comunicarse entre usuarios de UNIX. Existen también otras alternativas más de mensajes en tiempo real. Podríamos considerarlos el antepasado de los chats. Probemos un poco.

Mire el manual del comando `write` y pruébelo para mandar mensajes a otros usuarios en la misma máquina. El comando `wall` tiene una función parecida pero está pensado más para dar avisos generales a todos los usuarios del sistema, por ejemplo cuando el sistema va a apagarse se hace un `wall` para informar a todos los usuarios que se apagará en breve.

Un programa más sofisticado para hablar entre dos usuarios es el comando `talk`. Para hablar con un usuario en su mismo ordenador debe hacer:

```
$ talk usuario
$ talk usuario terminal
ejemplos:
$ talk rc21
$ talk rc10 pts/3
```

El usuario llamado recibe un mensaje y debe confirmar que quiere establecer la comunicación con otro `talk`.

Para probar el comando `talk` en el laboratorio debe tener en cuenta que el servidor de `talk` no está activado en todos los ordenadores. Está activado en algunos como `jupiter`, así que si quiere probar tendrá que acceder remotamente a él.

El servicio `ssh` lleva también integrado la posibilidad de transferencia de ficheros. Pruebe el comando `sftp` que es básicamente igual que el `ftp` pero utilizando una conexión encriptada como `ssh`. Si lo prefiere también puede usar el comando `scp` que permite hacer transferencia de ficheros pero con un formato parecido al `cp`. En lugar de hacer:

```
$ cp fichero_origen fichero_destino
```

Podemos hacer:

```
$ scp fichero_origen fichero_destino
```

Y tanto `fichero_origen` como `fichero_destino` pueden ser ficheros locales especificados como un camino o bien pueden ser ficheros en otro ordenador y se especifican diciendo `usuario@maquina:camino_al_fichero_relativo_al_home`.

Estableciendo conexiones

Recuerde que como se ha señalado en clase el comando `telnet` abre una conexión TCP con un servidor y nos permite enviar datos por la conexión y observar los datos que se reciben. Pruebe en el laboratorio a utilizarlo con esta finalidad.

Al estar conectado a Internet de forma indirecta no podrá llegar a los servidores web de Internet. Por ejemplo aunque haga:

```
$ telnet www.google.com 80
```

No conseguirá conectarse al servidor. Pero puede probarlo con servidores web que estén dentro de la red del laboratorio. Por ejemplo hay uno en `jupiter`. Pruebe a pedir una página manualmente con telnet a este servidor.

¡Un momento! Y si no funciona la conexión a www.google.com al puerto 80, ¿Por qué me funciona la página de google en el navegador?

Bueno de momento olvidémonos de este detalle, lo averiguará más adelante.

Utilice el comando `telnet` para obtener la fecha y hora del servidor de DAYTIME que está funcionando en `jupiter`. Para encontrar los puertos usados por las aplicaciones más comunes puede consultar la lista con los nombres de los servicios que mantiene el sistema en /etc/services. Nota: en ese fichero está la lista de los nombres de los servicios y sus puertos asociados, eso no implica que dicho ordenador tenga funcionando servidores para todos esos servicios.

Viendo el tráfico con `ethereal`

Vamos a ver los paquetes que se envían los PCs como resultado de los servicios que utilicemos. Para ello emplearemos el programa `ethereal`.

El programa `ethereal` nos permite observar los paquetes de red que son recibidos o transmitidos por un interfaz de red. Para ello lee del interfaz de red y muestra los paquetes recogidos, incluyendo en él el contenido de las cabeceras del paquete. Este tipo de programas se denominan sniffers. Esto se puede hacer en diferido, programando una captura y observando después los paquetes que han pasado, o bien en tiempo real viendo como aparecen paquetes conforme se ven en la red. Pero tenga cuidado una red como la del laboratorio puede transmitir miles de paquetes por segundo por lo que hacer a `ethereal` manejar muchos paquetes en la ventana mientras los captura puede consumir muchos recursos y hacer que reaccione de forma lenta. Utilice con cuidado la capacidad de tiempo real. En general es más fácil programar una captura y pararla para examinarla posteriormente. También es recomendable filtrar y capturar sólo los paquetes en los que estemos interesados.

Como veremos en clase, un interfaz de red conectado a una red de área local como la del laboratorio puede observar también paquetes dirigidos a otros ordenadores y no sólo los enviados por nuestro ordenador. Puede configurar el sniffer para que vea todos los paquetes posibles (a ésto se le denomina modo promiscuo) o bien para que sólo recoja los paquetes que envía o recibe nuestro ordenador.

Pruebe el `ethereal`:

Lance el programa `ethereal` (desde el menú o bien escribiendo `ethereal` en un terminal).

- Abra un terminal y haga:

```
$ ping jupiter
```

Deje el ping funcionando. El ping envía paquetes del protocolo ICMP que se transporta dentro de datagramas IP.

Inicie una captura con `ethereal`. Observe que mientras captura `ethereal` le muestra que está capturando paquetes de diversos protocolos. Cuando tenga algún paquete de ICMP que son los causados por el ping detenga la captura. Y observe los paquetes que ha capturado de la red. Busque los paquetes ICMP y observe qué dirección origen y destino llevan.

Puede indicarle al programa `ethereal` que filtre el tráfico que ve de forma que sólo muestre los paquetes ICMP. Para ello en la casilla de texto junto al botón *Filter* escriba *icmp*. Del mismo modo puede introducir este mismo filtro en la ventana de programación de la captura de forma que sólo capture los paquetes que cumplan el filtro.

El formato de estos filtros esta descrito en el manual de `tcpdump` otro programa sniffer en línea de comandos, predecesor de `ethereal`. Para ver el formato de estos filtros haga:

```
$ man tcpdump
```

Y busque en la descripción el formato de `expression`

Pruebe al menos y averigue que hacen los filtros siguientes (con el programa `ethereal`):

```
ip host 10.1.1.230
ip src 10.1.1.230
tcp port 80
tcp src port 80
```

Pruebe a capturar los paquetes de otros servicios.

- Prepare un filtro para capturar el servicio `telnet` y capture los paquetes de una conexión de telnet que realice con `jupiter`. Observe si puede ver los datos transmitidos dentro de los paquetes. ¿Puede ver la contraseña?

- Prepare un filtro y capture una petición web a `jupiter`. Observe si puede ver en los paquetes los comandos de HTTP

- Utilice `ethereal` para averiguar qué pasa cuando pedimos la página www.google.com en el navegador para que funcione. ¿Cómo obtiene el navegador la página?

Comando nc

En las prácticas siguientes aprenderemos a construir sockets clientes y servidores y a establecer conexiones y enviarse mensajes entre ellos. Para depurar estos programas, mientras los hacemos, necesitaremos poder establecer sockets clientes y servidores de prueba. Aparte de los servicios simples como DAYTIME, Linux nos ofrece una herramienta muy útil, el comando `nc`.

Consulte el manual sobre el comando `nc`. Observe que puede utilizar el programa `nc` para realizar estas cuatro funciones:

- Cliente TCP sencillo que permite interactuar con la conexión desde el teclado.
- Cliente UDP sencillo que permite enviar un datagrama UDP con cada linea que escribamos y muestra a su vez el contenido de los datagramas que llegan al puerto indicado.
- Servidor TCP que permanece a la escucha y acepta una conexión.
- Servidor UDP que permanece a la escucha y permite contestar a quién le escriba

Pruébelo realizando cada una de estas tareas:

- Utilice `nc` para pedir la página web de un servidor web del laboratorio `jupiter`.
- Utilice `nc` para pedir la hora al servicio DAYTIME de `jupiter`.
- Utilice `nc` para simular un servidor web en `jupiter`. Acceda remotamente a dicho ordenador desde su ordenador y elija un puerto `P`. A continuación lance el programa `nc` para escuchar conexiones en el ordenador elegido y el puerto `P`. Para comprobar que está escuchando abra un navegador web y pida la página:

```
http://jupiter:P/
```

Tendrá que enviar la página manualmente en el servidor, pero compruebe que el navegador se queda esperando por la página (y no da un error de página no encontrada) y dibuja la página que introduzca en el servidor. Con ésto habrá probado que puede usar `nc` como servidor TCP manual.

- Ponga un servidor TCP con `nc` en un ordenador y puerto y lance contra el una conexión con telnet:

```
$ telnet destino puerto
```

Observe como puede enviarse datos entre ambos programas. ¿Puede colocar el servidor de forma que sea suficiente para conectarse a él con hacer lo siguiente?

```
$ telnet destino
```

- Utilice `nc` para enviar un paquete UDP. Por ejemplo al servidor DAYTIME de `jupiter` y obtenga la fecha de dicho servidor pero utilizando UDP.

- Demuestre que la hora la ha obtenido mediante UDP y no TCP capturando los paquetes entre su ordenador y `jupiter`.

- Utilice `nc` para establecer un servidor UDP y compruebe que puede enviarle paquetes UDP desde otro `nc` que actué como cliente.

Consultando los sockets del sistema

El comando de UNIX `netstat` permite la consulta de diversos parámetros y tablas del sistema relacionadas con la red. En nuestro caso nos interesa porque permite comprobar las conexiones (o sockets) abiertos por todos los procesos del sistema.

Consulte el manual de `netstat` y averigüe cómo listar todos los sockets del sistema con su estado. Averigüe cómo listar sólo los TCP y solo los UDP.

Pruebe la opción `-n` que en éste y en muchos comandos parecidos hace que no se impriman los nombres de los puertos y las máquinas en forma de nombre, sino en forma numérica.

Lance un servidor TCP y otro UDP en un puerto que elija, digamos 65432, y busque su socket en la lista de `netstat`. Observe el estado en que se encuentra. A continuación haga una conexión desde un cliente con `nc` a cada uno de ellos y vuelva a buscar los sockets en la lista del sistema. ¿Qué ha cambiado? ¿Qué diferencia hay entre el caso TCP y UDP? ¿Por qué?

Conclusiones

En esta sesión se ha practicado con las herramientas de manejo de la red de Linux y se han utilizado diversos servicios de red. Del mismo modo se ha explicado el funcionamiento de un sniffer y una herramienta de creación de conexiones y servidores TCP y UDP como `nc` que servirá en futuras prácticas para probar las aplicaciones de red que se realicen.

No es necesario entregar nada en esta práctica.