Instalación y configuración de un servidor de IRC

Diego Berrueta Muñoz, *berrueta@geocities.com* y José Alfredo Oslé Posadas, *aosle@pinon.ccu.uniovi.es*
Julio 1998

Este documento describe los pasos y el software necesarios para crear un servidor de IRC con Linux: el demonio de IRC, el bot de gestión del servidor y un programa cliente de IRC.

Índice General

1 Nota de los autores

Este documento es el resultado de un trabajo realizado para el curso "Administración de un sistema UNIX con Linux", impartido en el verano de 1998 en la Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo (*EUITIO*).

Los autores quieren aclarar que no son, ni pretenden ser, expertos en IRC. Por eso, este documento puede contener errores. Esperamos que sepan perdonarlos y nos los comuniquen.

Para obtener más información sobre Linux, no olvide visitar la página web del *Proyecto Lucas*, (`http://lucas.ctv.es/`) donde se encargan de traducir las guías de Linux al castellano, y la página de *INSFLUG* (`http://www.insflug.org/`), que coordina la traducción de los documentos *HOWTO* y *FAQ*).

2 Introducción

2.1 ¿Qué es el IRC?

El IRC (*Internet Relay Chat*) surgió como una ampliación del programa *talk*, tratando de superar sus limitaciones. Se trata de un sistema de conversación en tiempo real a través de redes de ordenadores y, por supuesto, de Internet. Las principales diferencias con el *talk* son las siguientes:

- Incorpora un protocolo mucho más elaborado, independiente de la plataforma.
- Posibilidad de conversaciones en las que intervengan más de dos participantes: el límite depende sólo de las posibilidades de los servidores.
- Organización de las charlas en forma de múltiples *canales*: por establecer un símil, los canales son *salones* donde se dan cita un cierto número de usuarios. Esta posibilidad se aprovecha generalmente para realizar canales temáticos, que son el punto de encuentro de personas con inquietudes parecidas.
- Posibilidad de conectar varias redes, para poner en comunicación un gran número de usuarios.
- Los usuarios pueden ocultar su identidad, lo que facilita conversaciones más espontáneas, pero también puede crear riesgos de seguridad.

El IRC nació hace diez años, cuando el finlandés Jarkko Oikarinen escribió el primer programa para poder efectuar conversaciones con una estructura cliente-servidor. Su expansión fue muy rápida, puesto que pronto se implantó en la red nacional finlandesa (*FUNET*), y posteriormente en Noruega, Suiza y Estados Unidos.

La primera gran red de IRC fue *EFNet* (1990), y después surgieron otras como *Undernet* (`http://www.undernet.org`), *IRCNet*, *DALNet*, *StarChat*, *StarLink*, *NewLet*. También existen redes para las personas de habla hispana, como *RedLatina*, *IRC-Hispano* (`http://www.irc-hispano.org`) o la más reciente, *Esnet* (`http://www.esnet.org`). Algunas de estas redes son de grandes magnitudes: por ejemplo, en *Undernet* existen más de diez mil canales.

2.2 ¿Qué es un servidor de IRC?

Como se ha comentado, el IRC se basa en redes de ordenadores. En estas redes, al menos un de los ordenadores trabaja como *servidor*, y sus funciones son recoger todos los mensajes enviados por los usuarios y reenviarlos. Por tanto, los mensajes no son enviados directamente del emisor al destinatario, sino que pasan por el servidor.

Esta filosofía de funcionamiento permite al administrador del servidor ejercer un cierto control: por ejemplo, puede impedir el acceso a determinados usuarios, ampliar las posibilidades de conversación comunicando varios servidores o limitar las posibilidades de los usuarios. Además, hace que todos los usuarios se conecten, en principio, en igualdad de condiciones, excepto el administrador, por supuesto.

2.3 ¿Qué necesita un servidor de IRC?

Para que un sistema UNIX pueda hacer las funciones de un servidor de IRC, es suficiente con lo siguiente:

- Tener acceso a la red.
- Instalar el software apropiado.

Dando por supuesto el primer apartado, vamos a puntualizar más en el segundo. ¿Qué tipo de software se requiere? En principio, para el establecimiento de un servicio básico, es suficiente con ejecutar un *demonio* de IRC, que se encargará de las tareas descritas en el apartado anterior. Sin embargo, esto proporcionará un servicio precario: para un gestión eficiente y automática de la red de IRC, se hace necesario un *bot* o *script*, que se encargue de la gestión de usuarios y canales sin intervención por parte del administrador.

Con el fin de poder monitorizar y administrar la red, es conveniente que el servidor disponga también de un software cliente de IRC.

2.4 Comandos básicos de IRC

Antes de describir los comandos, es necesario conocer cuál es, a grandes rasgos, la filosofía del funcionamiento del IRC. Como ya ha quedado dicho, las conversaciones se organizan en canales, que a su vez se identifican por un nombre (generalmente comienza con el símbolo `&`), y un *topic*, que es una breve descripción de los temas a los que está dedicado el canal.

Para poder dar coherencia a las conversaciones, los usuarios también están identificados por un nombre, que se llama *nick* o *apodo*. Con el fin de poder mostrar una información más explícita sobre el usuario, muchas redes exigen introducir el nombre completo y dirección de correo electrónico para poder acceder; sin embargo, la mayoría de usuarios aprovecha estos campos para introducir algún comentario más o menos gracioso.

Cuando un usuario está dentro de un canal, cualquier texto que escriba y que no sea reconocido como un comando, será enviado a todos los demás usuarios que forman parte del canal, y se indicará el apodo del usuario emisor. De la misma forma, la recepción de mensajes es automática; simplemente aparecen en la pantalla cuando son enviados.

Dentro de un canal existe al menos un usuario que tiene más privilegios que el resto. Se conoce como *operador* (*Op*), y tiene, entre otros, el poder de expulsar a otros usuarios del canal.

Un usuario puede estar presente en varios canales, aunque para poder disfrutar plenamente de esta posibilidad, es necesario tener un software cliente potente y manejable.

Los comandos que se van a describir ahora son los más elementales que debe conocer un usuario para participar en las conversaciones. Estos comandos son enviados por el cliente al servidor, quien se encarga de reaccionar de forma adecuada ante ellos.

- `/HELP`: permite obtener ayuda.
- `/JOIN &nombre_canal, /CHANNEL &nombre_canal`:

  introduce al usuario en el canal especificado. Si este canal no existe, se crea (la gestión dinámica de canales es otra de las posibilidades del IRC), y además confiere al usuario el privilegio de ser el operador del canal.
- `/WHO nombre_canal`: muestra una lista de los usuarios del IRC. Si se indica un nombre de canal, restringe el listado a las personas que están presentes en ese canal.
- `/WHOIS nick`: muestra más información sobre un usuario del que conocemos su nick. Sin embargo, por las razones ya comentadas, la información no suele ser cierta.
- `/TOPIC cadena`: cambia el *topic* o descripción del canal (si tenemos privilegios para poder hacerlo).
- `/NICK nuevo_nick`: permite cambiar nuestro nick o apodo.
- `/MSG nick mensaje`: envía un mensaje personal a un usuario determinado, sin que los demás usuarios lo vean en su pantalla.
- `/SIGNOFF, /SI, /BYE, /EXIT, /QUIT`: cualquiera de ellos sirve para abandonar la conversación. Si se añade un texto detrás del comando, será enviado como mensaje de despedida.
- `/USERS`: muestra una lista de todos los usuarios conectados al servidor.
- `/AWAY`: se utiliza para anunciar que, sin abandonar el servidor, vamos a estar unos minutos sin atender a la conversación.
- `/QUERY nick`: invita al usuario indicado a participar en una conversación privada. Si el receptor responde con un comando similar, el diálogo particular dará comienzo.

Los siguientes comandos los utilizan exclusivamente los operadores:

- `/OPER nick password`: da al usuario indicado los privilegios de operador (si la contraseña es válida).
- `/KICK nick`: expulsa al usuario indicado del servidor.
- `/QUOTE comando`: permite enviar comandos al servidor. Se puede utilizar para matar el demonio o realizar operaciones privilegiadas.

Para una lista más completa de los posibles comandos de IRC, se puede acudir a *ftp://ftp.rediris.es/docs/rfc/14xx/1459*.

2.5 Ataques en el IRC

Como veremos más adelante, tanto el *demonio* de IRC como el *bot* de gestión de los canales dedican una parte importante de su funcionalidad a impedir la entrada de algunos usuario en las redes de IRC. ¿Por qué es tan importante poder hacer esto?

La razón es que existen usuarios con grandes conocimientos sobre el IRC, y que muestran una actitud beligerante respecto a las más desprevenidos, los usuarios noveles. Aprovechándose de las grietas del sistema, son capaces de expulsar a otros usuarios e incluso *colgar* máquinas remotas, mediante diversos tipos de estrategias:

- *Flood*: consiste en requerir mucha información de un usuario de manera continua (mediante `version, whois, ping, dcc`, etc.), con el objeto de que la máquina atacada devuelva todas las peticiones, llegando a poner en peligro de saturación al servidor, que procederá a expulsarla.
- *Floodnets*: son grupos de usuarios coordinados mediante *scripts*, que consiguen hacer el trabajo de *flood* más efectivo.
- *Botnets*: se trata de una versión automatizada (mediante *scripts*) del *flood*.
- *ICMP*: este tipo de ataques son mucho más sofisticados y malignos. Como dato, baste decir que existe un *script* escrito en sólo cinco líneas de *perl* y capaz de bloquear una máquina Windows remota.
- *Smurf*: consiste en enviar un *ping* a la dirección *broadcast* de una gran red, con la identidad del remitente falseada para que apunte a la persona atacada. Las respuestas de las máquinas de la red irán directamente a esta persona, rompiéndole la conexión.

Al crear un servidor de IRC, se debe poner mucha atención en la prevención de ataques. Los ataques por saturación son relativamente fáciles de evitar mediante un *bot* capaz de detectarlos sobre la marcha y anularlos. Sin embargo, contra los ataques *ICMP*, la mejor manera de defenderse es utilizar la última versión del núcleo y los parches.

Para dificultar aún más el seguimiento de los atacantes, éstos suelen emplear *bouncers*, que son demonios colocados en una tercera máquina y que se limitan a redirigir todos los paquetes que les lleguen. De esta forma, el pirata puede actuar sin temor a que descubran su IP.

Se puede consultar más información sobre estos temas (ataques, defensas y monitorización) en las siguientes direcciones: *http://www.irchelp.org/irchelp/nuke/* y *http://www.esi.us.es/~roman/tacticas.html*.

3 Demonio de IRC: `ircd`

3.1 Función de un demonio `ircd`

En una arquitectura centralizada como es la *arquitectura cliente-servidor* se distinguen precisamente estas dos partes: *cliente*, que es el que demanda un servicio; y el *servidor*, que es el que lo oferta. En la máquina servidor coexistirán tantos programas esperando a la escucha de nuevas peticiones como servicios diferentes se oferten.

Estos programas, denominados *demonios* (*daemons* en la terminología inglesa) son programas que están en ejecución, cargados en memoria, y que están encargados de responder a las peticiones que hacen los programas clientes, que generalmente están ubicados en otras máquinas, pero que pueden ejecutarse también en la propia máquina servidor. Por tanto, siempre hay una correspondencia directa entre el programa o demonio del servidor y el programa cliente de la máquina cliente.

En nuestro caso particular de IRC, el servidor será la máquina que ejecute el demonio *ircd* y los clientes serán los programas que los usuarios utilicen para *engancharse* a este servidor.

3.2 Descripción de `ircd-hybrid`

A la hora de instalar un servidor de IRC hay que elegir un demonio de `ircd`. Existen algunos como el `irdu`, utilizado en la red *Undernet*, o el más básico, `ircd`, sin modificación alguna. Pero nosotros hemos escogido el `ircd-hybrid` versión `5.2p1` por ser el más completo. Es el usado en la red *Efnet*.

3.3 Obtención e instalación

Algunas distribuciones como *Debian* incluyen este software entre sus paquetes básicos. Si utilizamos otra distribución, se puede recurrir al FTP o contactar con sus autores en la dirección *ircd-hybrid@vol.com*.

El primer paso es descomprimir el paquete con el comando:

```
tar xvzf ircd.tar.gz
```

Los ficheros descomprimidos los alojaremos en el directorio `/usr/src/ircd-hybrid-5.2p1/`.

Algunos de los ficheros más importantes son:

- `INSTALL`: describe los pasos a realizar para la instalación.
- `README.FIRST, README.hybrid, README.small_nets`: como siempre, conviene leerlos antes de proceder a la instalación.
- `Opers.txt`: describe los comandos para el operador del servidor IRC.
- `Doc/`: este directorio contiene documentación sobre IRC.

3.4 Configuración previa a la compilación

Como sugiere la documentación, es preciso revisar algunos ficheros donde se definen varios parámetros, por lo que será necesario compilar el programa. Sin embargo, el primer paso es adaptar los ficheros de configuración y compilación a nuestro sistema. Esto lo haremos ejecutando el *script* `./configure`.

A continuación, editaremos parte del código fuente (por motivos de optimización): puesto que nuestra red no va a tener las pretensiones de afluencia de usuarios que tiene *Efnet*, acortaremos un poco los recursos reservados por el *ircd* para que consuma menos memoria.

El primer fichero que cambiaremos será `include/config.h`, que define muchas de las directrices de comportamiento del servidor. Modificaremos dos líneas *define* de esta forma:

```
#define HARD_FDLIMIT    256
#define INIT_MAXCLIENTS 150
```

También quitaremos la línea

```
#define DO_IDENTD
```

para que no chequee constantemente la identidad de todo aquel que entre, ralentizando los accesos al servidor.

El fichero `src/list.c` define el tamaño de algunas tablas mediante símbolos del preprocesador. Pondremos el valor 64 a las variables `LINK_PREALLOCATE` y `CLIENTS_PREALLOCATE`, lo que es suficiente para una red donde no se prevee un acceso mayor a 60 usuarios.

El último paso antes de la compilación es editar el fichero `Makefile` resultante de ejecutar el *script* `./configure`, para que la línea que especifica los parámetros que usará el compilador `gcc` quede de la siguiente forma:

```
# Default CFLAGS
CFLAGS= -g -O2 -DCPU=586 -m486
```

Los dos últimos parámetros son opcionales, y sirven para optimizar el código generado para máquinas *Pentium/AMD*.

3.5 Compilación

Una vez hechos todos los ajustes pertinentes, se puede proceder a la compilación propiamente dicha, con el comando:

```
make
```

La compilación transcurrirá durante unos minutos, tras los cuales, si no se ha producido ningún error, sólo faltará crear el directorio `/usr/local/ircd/`, donde se instalarán los ficheros ejecutables gracias a la orden:

```
make install
```

3.6 El fichero `ircd.conf`

Ahora se configura el servidor de IRC. Esto se hace editando el fichero `/usr/local/ircd/ircd.conf`, que se estructura en diferentes líneas, identificadas con una letra, y que contienen parámetros separados por el signo ':'. Algunas de las líneas más importantes son:

- `A`: información sobre los administradores:
  1. Compañía
  2. Localización
  3. Dirección de contacto
- `M`: información sobre el servidor:
  1. Nombre del host
  2. Dirección IP (opcional)
  3. Nombre del servidor IRC
  4. Puerto (generalmente, 6667).
- `K`: restricciones de acceso a usuarios o redes completas:
  1. Nombre del host.
  2. Horario.
  3. Nombre de usuario.
- `C` y `N`: conexiones a otros servidores de IRC.
- `R`: establece restricciones a través de un programa externo de validación.
- `D`: otra forma de impedir conexiones a redes enteras, mostrando un comentario.
- `E`: excepciones a la línea `K`.
- `F`: super-excepción (similar a la anterior, pero se salta los límites de clases de usuarios).
- `Y`: define una clase de usuario:
  1. Número de clase.
  2. Frecuencia de ping (segundos).
  3. Usuarios por IP.

4. Número máximo de enlaces.
5. Tamaño de la cola de envío.

- `I`: permite el acceso a algunas máquinas:
  1. Dominio o dirección IP.
  2. Password del cliente.
  3. No documentado (generalmente '*').
  4. No documentado.
  5. Número de clase asociada.
- `o`: define operadores locales (recomendado):
  1. usuario@host.
  2. Clave encriptada.
  3. *Nick* del operador.
  4. No documentado (generalmente 0).
  5. No documentado (generalmente 0).
- `O`: define operadores globales.
- `P`: configura puertos adicionales.

Tras las modificaciones oportunas, nuestro fichero `ircd.conf` es:

```
M:maquina.euitio.uniovi.es:156.35.98.138:IRC EUITIO:6667
A:Universidad de Oviedo EUITIO:Asturias:SysAdm <root@maquina.euitio.uniovi.es>
Y:0:90:3:2:100000
Y:1:90:3:20:100000
I:*.euitio.uniovi.es::*::1
I:156.35.53.*::*::1
Y:3:90:1:100:100000
I:NOMATCH::*@*PPP*::3
I:NOMATCH::*@*slip*::3
o:root@maquina.euitio.uniovi.es:mAJif8plpvVls:Operador:0:0
# Fin del fichero
# No especificamos nada en las líneas
# - C, N, L y N: porque no nos conectamos a ningún otro servidor
# - K, R, D, E y F: porque no establecemos restricciones tan fuertes
# - P: porque no tenemos más puertos
```

Nota: para obtener el campo *clave encriptada* de la línea `o`, hay que utilizar la herramienta `mkpasswd`, ubicada en el directorio `tools`, que toma una cadena de texto y la devuelve encriptada.

Podemos verificar la correcta sintaxis del fichero de configuración `ircd.conf` mediante la utilización de la herramienta `tools/chkconf`.

También podemos modificar el fichero `ircd.motd` para establecer el mensaje del día que aparecerá a los usuarios que establezcan contacto con el servidor.

3.7 Ejecución del demonio

Por motivos de seguridad, no es posible lanzar el demonio como `root`, así que, o bien se lanza a mano como usuario `irc`, simplemente con:

```
/usr/local/ircd/ircd
```

o añadiendo la siguiente línea en el fichero `/etc/inetd.conf`:

```
ircd stream tcp wait irc /etc/ircd ircd -i
```

para que arranque automáticamente cada vez que se inicie el equipo servidor. Sin embargo, este segundo método no lo hemos podido confirmar porque, aparentemente, no ha funcionado.

4 El *bot* de gestión del servidor

4.1 Función de un *bot* de gestión del servidor

La función de un *bot* o *script* de gestión del servidor es ampliar las posibilidades del IRC, y a la vez permitir un cierto control sobre los usuarios. El *bot* se introduce como un usuario más, pero adquiere el poder del operador, lo que le permite administrar de manera eficiente los canales. Entre sus posibilidades están:

- Envío de mensajes periódicos a ciertos canales.
- Impedir la entrada de usuarios no deseados.
- Reconocer a los administradores del servidor y cederles el poder.
- Aumentar la seguridad general del servidor.
- Impedir que los usuarios puedan expulsarse entre ellos.

Existen multitud de *bots* disponibles. Entre los más importantes, podemos destacar:

- *Eggdrop*: posiblemente el más conocido, pero también el más complejo de instalar y mantener. Requiere TCL.
- *Uworld*.
- *Cservice*.
- *Argobot*.

4.2 *Argobot*

El *Argobot* es un *bot* relativamente sencillo escrito por el español Jesús Cea Avión (*http://www.argo.es/~jcea*), aunque algunas grandes redes hispanas de IRC están adoptándolo. Soporta todas las características antes enunciadas, pero sin profundizar en ellas, de forma que no resulta tan grande y complejo como *Eggdrop*. Como ventaja añadida, ocupa muy poca memoria y es muy eficiente. Es un *bot* multicanal capaz de:

- Proteger los modos de un canal.

- Dar y quitar los privilegios de operador de forma manual y automática.
- Enviar mensajes (*notices*) a los usuarios recién llegados a un canal.
- Controlar el acceso y configuración de modos y permisos independientemente para cada canal.
- Acceder a redes que exigen claves.

Otras características del Argobot son:

- Se distribuye con el código fuente completo (escrito en lenguaje C), y que además puede compilarse bajo UNIX o en plataformas Windows.
- Utiliza un fichero de configuración.
- Lleva registros (ficheros *log*) de toda la actividad que se produce en el servidor.
- Puede reconfigurarse sin necesidad de detener su actividad, es decir, puede modificarse su configuración y cargar de nuevo el *bot* sin que los usuarios adviertan ningún cambio ni interrupción del servicio.

4.3 Instalación

El *Argobot* se puede conseguir en la página web de su creador (*http://www.argo.es/~jcea/irc/argobot.htm*). Existen multitud de versiones, algunas desarrolladas específicamente para alguna gran red, y además se pueden conseguir una veintena de *parches* que corrigen algunos defectos y añaden nuevas características.

Si optamos por obtener el *Argobot* por Internet, descubriremos que apenas incluye unos pocos ficheros (código fuente C), y como única documentación un fichero `README` muy breve. Para paliar esta deficiencia, su autor a puesto toda la documentación disponible a través de Internet, en forma de páginas web. Es necesario, por tanto, descargar estas páginas y leerlas detenidamente.

El *Argobot* se suministra en forma de un fichero `.tgz`, que una vez descomprimido (en nuestro caso, en el directorio `/usr/src/argobot/`) da lugar a cuatro ficheros:

- `argobot.c`: fichero principal del código fuente.
- `argo_parser.c`: código fuente que interpreta el fichero de configuración.
- `argobot.conf`: fichero de configuración.
- `argobot.log`: mantiene un registro de los comandos enviados al bot.

Es importante comprobar que este último fichero se ha creado realmente. Debido a algunas configuraciones de `tar`, es posible que no se genere este fichero (porque está vacío), lo que provocará errores al intentar ejecutar el *Argobot*. En este caso, es necesario crear el fichero manualmente (con la orden `touch`, por ejemplo), y darle los permisos apropiados.

Nota: si queremos utilizar una de las características más avanzadas del *Argobot*, como es la propagación de líneas `K` (sólo tiene sentido en redes de varios servidores), será necesario seguir las instrucciones relativas a los permisos y el fichero `ircd.conf`.

4.4 Edición del código fuente

Antes de compilar, es conveniente comprobar que el código fuente esta bien adaptado a nuestro sistema. Las modificaciones que debemos hacer dependen de la forma en la que vamos a ejecutar el *bot*:

- Si vamos a ejecutar el *bot* como *root* (es el método recomendado), nos aseguraremos de que está definido `SEGURIDAD` (con la sentencia `#define SEGURIDAD`), y que las macros `UID` y `GID` tienen valores apropiados. Si además vamos a hacer un *CHROOT* (que es una medida adicional de seguridad), definiremos el directorio que debe tomar como raíz con la sentencia `#define CHROOT <directorio>`.
- Si vamos a ejecutar el *bot* como un usuario distinto a *root*, debemos eliminar el símbolo `SEGURIDAD` (con la sentencia `#undef SEGURIDAD`).

Además de estas modificaciones, podemos definir el símbolo `VERBOSE` para que se impriman en la pantalla todos los mensajes que el servidor envía al *Argobot*. Otro símbolo interesante es `CONTROL_FLOOD`, que por defecto está activado, pero que deberemos eliminar si el nodo al que conectamos está preparado para controlar los ataques por *flood*.

4.5 Compilación

Como ya se ha indicado, el *Argobot* puede ser compilado bajo múltiples plataformas. En Linux, es suficiente con hacer:

```
gcc -Wall -g argobot.c -o argobot
chown root argobot
chgrp root argobot
```

La primera instrucción realiza la compilación, generando el fichero `argobot`. Con las dos siguientes, establecemos los propietarios de este fichero.

4.6 Configuración

La tarea de configuración se limita al fichero `argobot.conf`, aunque también es necesario crear una nueva cuenta de usuario para el operador del servidor IRC.

El fichero `argobot.conf` es un fichero de texto que contiene líneas con comandos. Al hacer la instalación, se genera un fichero de ejemplo, pero lo más conveniente es modificarlo para adaptarlo a nuestras necesidades. Los comandos son:

- `IRCnick <nick>`: indica cuál es el *nick* que debe emplear el *bot* para identificarse como operador ante el nodo IRC.
- `IRCpasswd <password>`: complementa al comando anterior, indicando la clave necesaria para adquirir los privilegios de operador.
- `nick <nick>`: especifica el *nick* bajo el que aparecerá el *bot* a los usuarios.
- `server <servidor>`: sirve para indicar el servidor a que va a conectarse.
- `port <puerto>`: indica el puerto al que se va a conectar (generalmente, el 6667).
- `passwd <password>`: clave que será enviada al servidor al principio de la conexión, con lo que se podrá acceder a otras redes que necesiten claves.

- `away <mensaje>`: mensaje que se muestra a los usuarios al entrar.
- `nick_collide <mensaje>`: mensaje enviado a aquellos usuarios que estén utilizando el *nick* indicado en el comando `nick`. Inmediatamente después, se les expulsará mediante un *kick*.
- `umbral_kline <valor>`: indica cuántos intentos de conexión se permitirán antes de poner una *línea k* local. El valor más corriente es cinco. Esto es una medida de protección contra los usuarios que tengan una actitud sospechosa.
- `timeout_klines <valor>`: indica cuántos segundos permanecerá activa una *línea k* local. El valor más aconsejable es en torno a 900 (es decir, 15 minutos).
- `timeout_whowas <valor>`: indica cada cuántos segundos hay que comprobar la presencia de *clones* (usuarios que están presentes bajo varios *nicks*, lo que suele ser síntoma de actividades peligrosas). Es conveniente un valor en torno a cinco.
- `timer <offset> <periodo> <comando>`: se utiliza para enviar comandos periódicos al servidor de IRC, que se repetirán cada cierto número de segundos. El *offset* indica el retraso del primer envío.
- `set <alias> <máscara> [clave]`: define un usuario. El nombre de referencia (interno al *Argobot* será el *alias*, y se aplicará a aquel usuario que satisfaga las condiciones de la *máscara*, que tiene la forma `nick!usuario@dominio` (se admiten comodines).
- `group <nombre_grupo> <alias> [<alias>...]`: define un grupo de usuarios, cuyo nombre será el indicado, y al que pertenecerán los usuarios indicados a continuación.
- `join <canal> [clave]`: indica al *bot* que debe gestionar el canal indicado, entrando con la clave proporcionada, que es opcional. Si utilizamos la clave `GOD`, se forzará la entrada del *bot*. NOTA: dado que el carácter '#' se emplea para indicar comentarios, no debe escribirse en el campo *canal*. Por tanto, si escribimos `linux`, estamos refiriéndonos al canal `#linux`.

Puede haber tantos comandos `join` como sean necesarios. Para cada uno de ellos, se pueden indicar opciones específicas para el canal, mediante los siguientes comandos:

- `autoop <grupo>`: cualquier usuario del grupo indicado será automáticamente dotado de los privilegios de operador cuando entre en el canal.
- `privil <grupo>`: declara privilegiados a los miembros del grupo indicado, lo que les permitirá enviar comandos al *bot*.
- `mode_default <modos>`: indica el modo por defecto del canal.
- `mode_disallow <modos>`: prohibe algunos modos en el canal.
- `log <fichero>`: almacena toda la actividad del canal en un fichero. Se incluyen marcas temporales cada diez minutos.
- `notice <texto>`: indica un mensaje que será enviado a todos los usuarios que entren en el canal.
- `allow_any_ban`: permite a los operadores del canal hacer prácticamente todo, lo que no es muy aconsejable.

Para nuestro sistema, el fichero `argobot.conf` es el siguiente:

```
#
# Parámetros globales
#

# Nick utilizado a la hora de identificarse como IRCop, así como
# para el WHOIS
IRCnick ArgoBot

# Password correspondiente al nick anterior
IRCpasswd miclave
# Nick bajo el cual debe aparecer el bot
nick _ArgoBOT
passwd miclave

# Nombre del dominio al que se va a conectar el bot
server maquina.euitio.uniovi.es
port 6667

# Mensaje que aparece en el away del bot
away Bot de control de maquina.euitio.uniovi.es. No respondemos \
de los fallos.

# Mensaje enviado con el KILL a cualquier usuario que esté utilizando el
# nick definido en el comando NICK anterior
nick_collide Escoge otro Nick, por favor

# Las siguientes líneas configuran distintos aspectos de la seguridad
umbral_kline 5  # Máximo número de intentos (desconectado)
timeout_klines 15  # 15 minutos
timeout_whowas 5 # Tiempo entre whowas

#
#
# Grupos de usuarios
#
#

# Define el grupo de usuarios al que pertenecen todos
set todos *!*@*      # Comodines
group todos todos

# Define el grupo de IRCops
set diego *!diego@maquina.euitio.uniovi.es
set alfredo *!alfredo@maquina.euitio.uniovi.es
group IRCops diego alfredo

# Define el grupo de proveedores
group proveedores diego alfredo

# Canal ayuda-esnet
group ayuda-esnet diego alfredo

#
#
# Canales
```

```
#
#

join linux GOD
autoop IRCops
mode_default ntm
mode_disallow silpko
notice Canal dedicado a los amantes del Linux. \
Prohibido a Bill Gates.
timer 60 60 privmsg #linux :Mensaje enviado al canal linux de forma \
automática
timer 60 9999 topic #linux :Canal para los autenticos entendidos en \
informática y sistemas operativos.

# Pone un modo para reducir el consumo de ancho de banda
timer 60 600 mode _argobot :+d
```

4.7 Ejecución

Para ejecutar el *bot*, basta con escribir (preferiblemente como *root*):

```
cd /usr/src/argobot
./argobot
```

Si iniciamos *Argobot* de esta manera, imprimirá por la pantalla toda la actividad que registre (como entradas y salidas de usuarios), lo que puede ser conveniente por motivos de depuración, pero resulta innecesario durante una ejecución normal. Por tanto, es muy frecuente ejecutar el *Argobot* en segundo plano y redireccionando la salida:

```
nohup argobot &
```

Esto nos evitará ver todos los mensajes por la pantalla, pero los enviará a un fichero llamado `nohup.out` situado en el directorio actual. El problema es que este fichero puede crecer muy rápidamente, de manera que en grandes redes se suele enlazar con el dispositivo nulo, para convertirlo en un *agujero negro*:

```
ln -s /dev/null nohup.out
```

Durante la ejecución de una sesión de IRC, los usuarios autorizados pueden enviar mensajes al *bot* para cambiar su funcionamiento. Para realizar esta tarea, se emplea el comando `MSG` del IRC:

```
/MSG _argobot <clave> <comando> <canal> [nick]
```

Donde *comando* puede ser:

- `op`: da privilegios de operador al usuario indicado.
- `deop`: quita los privilegios de operador al usuario indicado.
- `invite`: invita a un usuario a un canal.

5 Cliente de IRC

5.1 Función de un cliente de IRC

La función de un cliente de IRC es conectarse al servidor y filtrar la información disponible para cada usuario. Además, debe encargarse de enviar los mensajes y comandos que escribe el usuario mediante el protocolo de IRC.

El cliente habitualmente reside en la máquina del usuario y se conecta al servidor a través de un puerto (generalmente el 6667). Existen clientes de IRC para todas las plataformas, dado que no es necesario que la plataforma cliente coincida con la del servidor. En los sistemas Windows, el cliente más popular es el *mIRC*, que nosotros elegimos para hacer las primeras pruebas y asegurarnos de que el *demonio* estaba funcionando correctamente. Una vez confirmado el buen funcionamiento del servidor, instalamos un cliente para Linux. Para esta plataforma, la oferta es muy grande. Los más conocidos son:

- *Kirc* (*ftp://ftp.kde.org/*): el más antiguo cliente en el entorno *KDE*. Tiene un entorno gráfico muy cuidado, pero carece de algunas características básicas.
- *Ksirc* (*http://www.ksirc.org/*): también para el *KDE* y resultado de la evolución de un cliente en modo texto (el *Sirc*). Está todavía en fase de desarrollo.
- *Keric* (*http://www.indonesia-undernet.org/keirc*): otro más que funciona bajo *KDE* y es uno de los más prometedores, aunque aún está muy lejos de poder ofrecer una versión completamente estable.
- *BitchX* (*http://www.bitchx.com/*): el cliente en modo texto preferido por los usuarios de Linux.
- *ScrollZ* (*http://www.scrollz.com/*): tiene características similares al *BitchX*, y es el más veterano de todos, lo que le permite alcanzar una buena velocidad y un gran nivel de optimización (menor consumo de recursos).
- *YagIRC* (*http://www.sicom.fi/~ikioma/yagirc.html*). Cliente para el entorno *GNOME*, bastante reciente y capaz de mostrar un interfaz en modo texto o en modo gráfico.

5.2 Descripción de *BitchX*

Entre sus características están:

- Varias conversaciones en el mismo canal.
- Múltiples servidores.
- Autocompletado de *nicks*.
- Soporte para *scripts*.
- Cliente de FTP incorporado.
- Nuevas características exclusivas, como son: cliente de correo integrado y un reproductor de CDs musicales.

5.3 Obtención e instalación

El *BitchX* puede conseguirse en *http://www.bitchx.com/*, desde donde se pueden conseguir varios ficheros: uno de ellos contiene el paquete básico y los demás son accesorios opcionales.

Tras descomprimir y desempaquetar el fichero con la orden *tar* en el directorio `/usr/local/BitchX/`, obtendremos directamente los ejecutables, puesto que se distribuye con los fuentes ya compilados.

5.4 Configuración

No contiene ningún fichero de configuración, sino que las opciones se establecen mediante variables de entorno. Las principales son:

- `IRCNICK`: establece el *nick* por defecto.
- `IRCUSER`: establece el nombre de usuario por defecto.
- `IRCNAME`: permite escribir el nombre real del usuario.
- `IRCSERVER`: nombre del servidor a utilizar.
- `IRCPORT`: puerto a utilizar.

Es conveniente dar valor a estas variables en los *scripts* de arranque o bien en el fichero `~/.profile` de cada usuario del sistema.

5.5 Ejecución

El ejecutable se llama `BitchX-tcl`, pero resulta conveniente crear un enlace simbólico llamado `irc` situado en un directorio que forme parte del *path* para simplificar el trabajo de los usuarios.

6 Anexo: El INSFLUG

El *INSFLUG* forma parte del grupo internacional *Linux Documentation Project*, encargándose de las traducciones al castellano de los Howtos, así como de la producción de documentos originales en aquellos casos en los que no existe análogo en inglés, centrándose, preferentemente, en documentos breves, como los *COMOs* y *PUFs* (**P**reguntas de **U**so **F**recuente, las *FAQs*. `:)`), etc.

Diríjase a la sede del Insflug para más información al respecto.

En ella encontrará siempre las **últimas** versiones de las traducciones «oficiales»: *www.insflug.org*. Asegúrese de comprobar cuál es la última versión disponible en el Insflug antes de bajar un documento de un servidor réplica.

Además, cuenta con un sistema interactivo de gestión de fe de erratas y sugerencias en línea, motor de búsqueda específico, y más servicios en los que estamos trabajando incesantemente.

Se proporciona también una lista de los servidores réplica (*mirror*) del Insflug más cercanos a Vd., e información relativa a otros recursos en castellano.

En *http://www.insflug.org/insflug/creditos.php3* cuenta con una detallada relación de las personas que hacen posible tanto esto como las traducciones.

¡Diríjase a *http://www.insflug.org/colaboracion/index.php3* si desea unirse a nosotros!.

«Cartel» Insflug, *cartel@insflug.org*.