INSTALACION

DE UN

SERVIDOR

DE CORREO SEND MAIL

Índice

Introducción al sendmail

El servicio de transferencia de correo +electrónico entre máquinas pertenecientes a Internet es de los más antiguos que se vienen ofreciendo en la historia de esta gran red. También es, quizás, el más representativo de la filosofía cliente servidor dentro de la extensa familia de protocolos TCP/IP.

El servicio de correo electrónico permite llevar a cabo el envío y recepción de mensajes con usuarios de otros ordenadores de la red. El modo más sencillo de implementar este servicio es mantener ficheros de correo, llamados buzones, en cada uno de los sistemas en los que se pretende intercambiar mensajes con alguno de sus usuarios.

Este sistema se basa en un método que permite añadir un mensaje a esos ficheros, o bien recuperarlos. El problema principal aparece cuando los ordenadores entre los que se intercambia correo no tienen continuamente activo el gestor de correo.

Esto se solventa mediante la instalación de mail servers, que es lo que yo he realizado. Voy a explicar un poco como funciona la gestión y distribución clásica de este tipo de mensajes.

1.1. Cliente, servicio, servidor y protocolo

El cliente e-mail es un programa habitualmente llamado "*mail*" que permite la lectura de los ficheros buzón a cada usuario autorizado. También es capaz de manejar el "*spool*" de correo, es decir, de enviar mensajes que serán leídos por el servidor e-mail para poder ser distribuidos a otros usuarios.

El servicio encargado de proporcionar el intercambio de correo electrónico entre las máquinas se denomina servicio de e-mail, y puede ser ofrecido por distintos programas daemon, de los que el más extendido se llama "*sendmail*".

El daemon sendmail es el servidor de correo electrónico más popular entre las máquinas de Internet. Según algunos estudios, constituye entre el setenta y el ochenta por ciento de las implementaciones existentes hoy en día.

1.1.1. El sendmail

También se conoce con el nombre de "*delivery*", o bien distribuidor de correo. Realiza su función manteniéndose a la escucha del socket 25, comunicándose con los daemons de otros sistemas para recibir el correo entrante y enviar el correo saliente.

En cuanto a la aplicación TCP/IP, se utiliza el protocolo SMTP (Simple Mail Transfer Protocol), el cual se caracteriza por su eficiencia, sencillez y facilidad de depuración, gracias a los mensajes que acompañan a sus comandos.

1.1.2. Explicación básica de SMTP

La gestión de todo lo concerniente a la red por parte de IP y TCP deja a los programas de aplicación como un simple intercambio de comandos y datos. Por ejemplo, el protocolo de correo, del que podemos hacer un seguimiento muy sencillo, trabaja de la manera que vamos a describir a continuación.

Nuestro programa de mail abre una conexión contra el mail server remoto. Entonces, envía su nombre de máquina local, así como el nombre del emisor, el buzón de destino y un comando diciendo que empieza el texto del mensaje.

En este punto, el servidor finaliza el tratamiento de lo que ha asumido como comandos y comienza a aceptar el mensaje hasta que recibe una marca especial. Después de esto, ambos programas entienden que el envío de comandos ha sido retomado.

Un ejemplo de cómo se podría hacer sería de la siguiente forma, suponemos que enviamos un mensaje desde norba:

```
Date: Sat, 22 May 98 04:22:28 GMT
From: mbone@norba.indexado.com
To: root@Arraquis.com
Subject: Hola root

Este es el mensaje
```

El formato del mensaje está basado en Internet estándar (RFC 822), donde también se especifica que el mensaje debe transmitirse como net ASCII (ASCII con <cr><lf>), con unas líneas de cabecera determinadas, una línea en blanco, y, a continuación, el cuerpo del mensaje que deseamos enviar.

Los formatos estándares, diseñados para las direcciones, se han hecho mucho más flexibles

Las direcciones se indican de la forma clásica por todos conocida, como "*usuario@máquina*".

Éste era el formato que fue diseñado inicialmente, pero rápidamente los estándares se hicieron mucho más flexibles para contemplar casos de máquinas integradas bajo dominios, con la nomenclatura "*usuario@máquina.subdomino(s).dominio*", donde puede que una de las máquinas sea el servidor central de correo de las demás.

Por lo demás, también existe la posibilidad de que el campo "*usuario*" no sea realmente un usuario del sistema. Esto permite manejar listas de dichos usuarios y nombres genéricos. En el ejemplo que estamos desarrollando, la primera operación consiste en preguntar quién maneja el correo para norba.indexado.com.

El servidor de correo responde que norba.indexado.com maneja su propio correo. El programa local de correo en arrakis consulta la dirección de mbone.norba.indexado.com, (o al servidor de nombres si lo hubiera), y obtiene su dirección Internet. Entonces, abre una conexión al socket 25 (servidor SMTP) y tiene comienzo la conversación deseada.

El intercambio de mensajería, en lo que se refiere a la aplicación, será el siguiente que se enumera:

```
(arrakis) 220 arrakis.com SMTP Service at 22 May
98 05:02:28 GMT
(norba) HELO norba.indexado.com
(arrakis) 250 arrakis.com –Hello,
norba.indexado.com
(norba) MAIL From:<mbone@norba.indexado.com>
(arrakis) 250 MAIL accepted
(norba) RCPT To: <root.arrakis.com>
(arrakis) 250 Recipient accepted
(norba) DATA
(arrakis) 354 Start mail input; end with
<CRLF>.<CRLF>
(norba) Date: Sat, 22 May 98 04:22:28 GMT
(norba) From: mbone@norba.indexado.com
(norba) To: root@arrakis.com
(norba) Subject: Hola root
(norba)
(norba) Este es el mensaje
(norba) .
(arrakis) 250 OK
(norba) QUIT
(arrakis) 221 arrakis.com Service closing
transmission channel
```

Para poder reproducir todo esto, por ejemplo por medio de norba y arrakis, podemos suplantar al programa de mail de norba haciendo un telnet a arrakis, pero al puerto 25, escribiendo la orden siguiente:

```
telnet arrakis 25
```

A partir del momento en que hayamos escrito esta imprescindible línea, para arrakis nosotros ya nos habremos convertido en un programa cliente de correo que ha solicitado una conexión, y como tal nos va a tratar.

Los comandos utilizan texto normal, característica del estándar Internet, lo que resultará muy útil a la hora de ver qué es lo que está sucediendo y ser capaces de diagnosticar problemas.

Por ejemplo, el programa de mail mantiene un log de cada conversación. Si algo sale mal, este log puede ser enviado al administrador de correo, quien tiene la posibilidad de interactuar directamente contra el servidor SMTP.

En protocolos que sean un poco más complicados, este proceso no es práctico en absoluto y se utilizan formatos binarios. Sin embargo, suelen estructurarse como registros de C (struct) o bien de Pascal (record).

Otra característica del ámbito Internet es que las respuestas empiezan con un número definido en el protocolo, lo que evita ambigüedades.

El log del programa de mail, puede enviarse al administrador que interactua con el servidor SMTP

Este número es lo que realmente tiene efecto en la operación de los programas; el texto en realidad es meramente informativo. De entre estos números de respuesta, lo verdaderamente importante es el primer dígito:

Un número de respuesta que empieza por 2 indica una ejecución satisfactoria.

Los que empiezan por 3 requieren una acción a continuación. El rango de los que comienzan por 4 es de error temporal (por ejemplo cuando el disco se encuentra lleno), y el mensaje tendrá que ser salvado para reintentarlo más tarde. El 5 es error permanente, (por ejemplo, no existe recipiente), y el mensaje tendrá que ser devuelto al emisor con un código de error.

Cada sesión empieza con un HELO, que proporciona el nombre del sistema que ha iniciado la conexión. Después de eso, por medio de una sentencia MAIL será especificado el emisor del correo y un grupo de sentencia RCPT de los recipientes destino.

Finalmente, se envían los datos mediante DATA, acabándolos con una línea que contiene tan sólo un punto en su primera columna.

Si hubiese que incluir una línea de esas características como parte del mensaje, entonces habría que hacer que el punto fuera doble.

Después de que el mensaje haya sido aceptado por el receptor, el programa de mail tiene dos posibilidades: enviar otro mensaje o bien terminar la conversación mediante QUIT.

1.1.3. El cliente mail (recepción de correo)

La llamada a mail sin argumentos permitirá a un usuario leer el correo que se encuentre almacenado en su buzón.

Por defecto, los ficheros buzón de correo se almacenan en el directorio /usr/spool/mail con el login name de cada usuario. Una vez que los mensajes han sido leídos, desaparecen del buzón y se almacenan en el fichero *'mbox'* que se encuentra en el directorio de conexión del usuario. Para leer el fichero mbox, en vez del buzón correspondiente, se pude utilizar la opción –f.

La lectura de un buzón de correo presenta los mensajes perfectamente ordenados, cada uno con su número de mensaje, con objeto de poder ser accesibles de manera independiente.

Una vez seleccionado un mensaje se puede disponer del mismo para llevar a cabo las siguientes funciones:

1. Visualizar su contenido mediante 'p' (print) o, simplemente, pulsando <intro>.
2. Borrarlo pulsando 'd' (delete). Esta acción es reversible mediante el comando 'u' (undo).
3. Responderlo, es decir, mandar correo al emisor del mensaje, con el comando 'r' (reply).

Para poder salir de nuestra sesión con mail se utiliza la orden 'q' (quit) o bien la orden 'x' (Exit).

Esta última aborta la sesión con todos los cambios que hayamos realizado. Es decir, no efectúa nunca modificaciones sobre el propio buzón del usuario.

Lo explicado hasta ahora constituye lo necesario para una sesión habitual con 'mail' es bastante más completo.

Si queremos obtener mas ayuda sobre el mail, podemos referirnos a la ayuda del man, poniendo '*man mail*'. (En algunos sistemas, la ayuda se obtiene poniendo '*man mailx*' o '*help mail*').

1.1.4. El cliente mail (envío de correo)

Si queremos enviar un mensaje a uno o más usuarios que se encuentren en nuestra red, basta con utilizar el comando '*mail*' con los nombres de dichos usuarios como parámetros.

Se pueden dar algunas causas por las que un determinado mensaje no pueda ser distribuido, (por ejemplo, por un repentino error del sistema, o bien que el recipiente para determinado usuario no exista, etc.).

Para enviar un mensaje el parámetro utilizado es el nombre del propio usuario

En ese caso, el mensaje que haya sido escrito no llegará a perderse, sino que será almacenado dentro de un fichero llamado "dead.letter", que se encuentra situado en el interior del directorio HOME del usuario.

Durante el proceso de escritura de los mensajes que se quieren enviar, se puede utilizar el carácter de escape tilde (~) al principio de una línea, para hacer una llamada alguna de las funciones especiales que se encuentran a nuestra disposición. Si el juego de caracteres que estamos utilizando es el que se corresponde con el mapa "es.map".

Algunas de las funciones que pueden ser más útiles para nuestros fines son las que se muestran a continuación:

7~!comando – Ejecuta el comando indicado y retorna a la edición de correo.

7~e – Invoca a editor de textos establecido. Esta función puede ser muy útil cuando estemos manejando mensajes que sena especialmente extensos.

7 ~h – Edita para su modificación la información de cabecera del mensaje.

7 ~q – Aborta el envío del mensaje, copiando lo que se ha escrito hasta ese momento en el fichero "dead.letter". (Debe estar activa la opción '*save*').

7 ~d – Recupera el contenido del fichero "dead.letter" añadiéndolo al mensaje en curso.

7~wnomfich – Escribe el mensaje en el fichero "nomfich" y continúa con la edición.

1.1.5. Configuración del cliente mail

La activación y la desactivación de todas las opciones de comportamiento que posee el programa '*mail*' se pueden realizar a través de dos comandos, denominados "set" y "unset".

De todas maneras, no todo queda ahí. Se dispone, en efecto, de una segunda posibilidad que puede ser muy útil: también pueden establecerse si utilizamos convenientemente los ficheros de configuración.

En la configuración del cliente mail intervienen dos ficheros:

/etc/mail.rc
$HOME/.mailrc

/etc/mail.rc contiene una serie de opciones del cliente mail que pueden ser controladas mediante los comandos '*set*' y '*unset*'. De esta forma, se pueden activar y desactivar las distintas opciones booleanas.

Otras opciones se definen con variables que pueden adoptar distintos valores. La información completa sobre la composición del fichero '*mail.rc*' se pude obtener tecleando el comando '*man mailx*'.

Las opciones de comportamiento del cliente mail pueden ser establecidas particularmente para cada usuario a través del fichero '*.mailrc*', que se encuentra en los distintos directorios de conexión.


Los directorios y ficheros especificados son los de Linux, para el resto de sistemas operativos, habría que comprobar si coinciden.

1.2. Los RFC's para el sendmail

El sendmail está basado en los siguientes protocolos RFC821 (Simple Mail Transport Protocol), RFC822 (Internet Mail Format Protocol), RFC1123 (Internet Host Requirements), RFC1521 (MIME), RFC1651 (SMTP Service Extensions), RFC1891 (SMTP Delivery Status Notifications), RFC1892 (Multipart/Report), RFC1893 (Mail System Status Codes), RFC1894 (Delivery Status Notifications), y RFC1985 (SMTP Service Extension for Remote Message Queue Starting). Estas son todas las referencias a los protocolos que utiliza el sendmail, para cualquier modificación o simplemente entender como funciona el sendmail, habría que referirse a estos manuales, donde está descrito cada uno de los formatos, protocolos, y el porque de algunas funciones del sendmail.

Estudiar o explicar todos estos RFC (Request for Comments), sería ampliar demasiado lo que es la instalación del sendmail, ya hemos hecho una introducción al protocolo SMTP ya que es principalmente en lo que se basa el sendmail, hemos hablado de cómo funciona, y se establece la conexión para el envío de correo, hemos descrito algunos de los comandos básicos que posee el sendmail, y ahora pasaremos a lo que sería la instalación del sendmail.

2. Pasos previos a la instalación

Antes de explicar como instalar el sendmail, vamos a ver que es lo que poseemos, describiremos los ficheros que tenemos, y cuales serán los pasos previos para realizar la instalación.

El Sendmail al que nos vamos a referir es la versión 8.8.8, esta versión se pude obtener en diferentes sitios de Internet, destacar entre ellos:

ftp.rediris.es
ftp.sendmail.org
ftp.cc.Berkeley.edu

Esta versión se suele obtener comprimida primero con tar y después con gzip, luego lo primero que hay que hacer es descomprimir esta versión, al descomprimir el fichero tar se creará una estructura de directorios, donde nos encontraremos las siguientes cosas.

2.1. Documentación

 CHANGES-R5-R8

Este fichero nos indicará los cambios que se han producido desde la revisión 5 hasta la revisión 8 del sendmail. Las diferencias, los nuevos ficheros que se han incluido, etc...

 FAQ

Respuestas a preguntas frecuentes que se suelen hacer sobre el sendmail. En este fichero encontramos una recopilación de las preguntas más frecuentes que se suelen hacer, estas preguntas están sacadas de diferentes news sobre el sendmail.

 KNOWNBUGS

Aquí se encuentran los bugs que se conocen sobre la reciente versión, la actualización de este fichero se puede encontrar tanto en ftp.sendmail.org como en ft.cs.berkeley.edu.

 RELEASE_NOTES

Contiene una descripción detallada de los cambios en la versión actual, explicados con detenimiento a título informativo.

 SRC/READ_ME

Contiene los detalles sobre la compilación y la instalación del sendmail.

 CF/README

Contiene todo lo que hace falta saber sobre el fichero de configuración, como crearlo y cual es el contenido de este fichero.

 DOC/OP/OP.ME

Contiene la guía de instalación y operación del sendmail, tanto en formato texto, como en PostScript.

2.2. Rutinas de la base de datos

Como base de datos se recomienda que se utilice la versión 1.85, ya que la versión anterior contenía varios bugs importantes, la nueva versión se puede obtener en la siguiente dirección http://www.sleepycat.com/packages/db.1.85.tar.gz.

2.3. Servicio de nombres host

Si se utiliza NIS o /etc/hosts, es importante que el nombre que se especifica sea completo y se recomienda que se utilice el fichero de hosts para construir una base de datos NIS, las direcciones deben de especificarse completas, ejemplo:

128.32.149.68 mastodo.CS.Berkeley.EDU mastodon

y no

128.32.149.68 mastodon

Si no se lleva a cabo de esta forma, el sendmail tendrá problemas para acceder a las direcciones, no tomando el nombre como un nombre cualificado, y tomándolo como un nombre corto, produciéndose así mensajes de error.

2.4. Usándolo con MH

Esta versión del sendmail notifica y responde con ciertos mensajes de SMTP indicando violaciones que son ignoradas por versiones anteriores. Si estas ejecutando MH y en el path contrib/mh.patch prevendrás estos mensajes de error. Ya que este path trabaja con versiones viejas del sendmail.

2.5. Usando identificaciones

El sendmail 8 soporta el protocolo IDENT definido en el RFC 1413. No identificando servidores que no incluyan esta distribución. Se pueden encontrar copias en los siguientes puntos:

ftp.lysator.liu.se	/pub/ident/servers
romulus.ucs.uoknor.edu	/networking/ident/servers
ftp.cyf-kr.edu.pl	/agh/uciagh/network/ident

Si se ejecuta el sendmail en un servidor IDENT, se sugiere obtener una copia de una de estas direcciones, disponiendo de versiones para diferentes sistemas.

2.6. Ficheros de compilación

Los ficheros de compilación están creados para la versión 4.4BSD, Si se compila con otra versión habría que revisar el fichero de configuración para modificar ciertos errores que se pueden producir, para mas detalles referirse al fichero src/READ_ME

2.7. Permisos de directorios

Para solucionar problemas que se produjeron en versiones anteriores, sendmail comprueba algunos subdirectorios para determinar si estos están protegidos. Para ello se recomienda ejecutar el siguiente comando:

Chmod go-w / /etc /usr /var /var/spool /var/spoll/mqueue

Al inicializar la base de datos se producirán los siguientes mensajes de warning:

WARNING: writable directory /etc

WARNING: writable directory /usr/spool/mqueue

Esto indicará que estos directorios no están protegidos contra escritura y peligra la seguridad dejando un pequeño hueco de entrada al descubierto.

2.8. Estructura de directorios

La estructura de directorios que se obtiene al descomprimir el sendmail es la siguiente:

cf	Código fuente de los ficheros de configuración
contrib	Algunas utilidades de ayuda al sendmail, aunque no se asegura que no den problemas
doc	Documentación
Mail.local	El código fuente de la base de datos, el cual no pertenece al sendmail, luego debe de compilarse independientemente.
mailstats	Estadísticas impresas por los programas, donde se indicará cual ha sido el resultado de las compilaciones, y los cambios efectuados.
makemap	Un programa para la creación de mapas de llaves usados por la construcción del sendmail
praliases	Un programa que imprime la base de datos de la versión del fichero de aliases
rmail	Fuente para el rmail. Este se utiliza para liberar al agente de UUCP, se recomienda no instalar versiones anteriores del rmail con el sendmail 8
Smrsh	Contiene una shell con restricciones del sendmail, la cual se puede utilizar reemplazando en /bin/sh para proveer de controles de seguridad al sendmail.
Src	Código fuente del sendmail
Test	Algunos ficheros de scripts para verificaciones.

Ya podemos compilar nuestro sendmail

3. Iniciando el proceso de instalación

Para compilar el sendmail, este provee su propio fichero script de compilación, el cual se encarga de averiguar que arquitectura estamos utilizando, mediante el comando *uname*, y compilará el fichero correspondiente para esa arquitectura. Dentro del subdirectorio /src encontramos el fichero de compilación *makesendmail*, ejecutamos este shell tecleando *sh makesendmail*. Y el sendmail se compilará solo creando nuestro ejecutable *sendmail.*

El estudio de este fichero sería bastante complicado, ya que son muchas las variables de entorno, así como macros que utiliza, para aquel que le interese profundizar en el estudio de este fichero, debe referirse al fichero *READ_ME* [Ref1], que se encuentra en el subdirectorio /src, donde se explican las variables que posee el fichero y las definiciones que posee. También podemos referirnos al manual de instalación y operación del sendmail [Ref2], en el cual se explica como modificar este fichero y adaptarlo a las necesidades de cada uno.

Los ficheros fuentes que posee el sendmail son los siguientes:

7**Makefile** El makefile utilizado para la compilación; Esta versión solo trabaja con el nuevo make de Berkeley.

7**Makefile.dist** Otra versión del makefile que trabaja con las viejas versiones del make.

7**alias.c** Crea los nombres de los alias en todos los formatos.

7**arpadate.c** Una subrutina que crea el estándar ARPANET.

7**clock.c** Rutinas a implementar tiempo real orientado a funciones del sendmail – e.j. timeouts.

7**collect.c** Las rutinas que actualmente leen el mail en un fichero temporal.

7**conf.c** El fichero de configuración. Contiene información sobre el fichero de configuración, para cualquiera que sea su configuración.

7**convtime.c** una rutina para el proceso de tiempos.

7**daemon.c** Rutinas para implementar el modo daemon. Esta versión es especial para Berkeley 4.1 IPC.

7**deliver.c** Rutinas para liberar mail.

7**domain.c** Rutinas para el interface con DNS (Domain Name System).

7**err.c** Rutinas para imprimir los mensajes de error.

7**envelope.c** Rutinas para manejar las estructuras del correo.

7**headers.c** Rutinas para procesar las cabeceras de los mensajes.

7**macro.c** La macro expander. Esta es usada internamente para insertar información desde el fichero de configuración.

7**main.c** Las rutinas del cuerpo del sendmail. Este fichero contiene rutinas variadas.

7**map.c** Soporte para mapas de bases de datos.

7**mci.c** Rutinas que manejan las conexiones al mail con la información guardada.

7**mime.c** Rutinas de conversión MIME.

7**parseaddr.c** Las rutinas para parchear las direcciones.

7**queue.c** Las rutinas que implementan los cuerpos de los mensajes.

7**readcf.c** Las rutinas que leen el fichero de configuración y lo transforman al formato interno.

7**recipient.c** Rutinas que manipulan la lista de recipientes.

7**safefile.c** Rutinas que chequean los modos o permisos de los ficheros, cuando se abren o crean ficheros.

7**saemail.c** Rutinas que salvan los mensajes en procesos de error.

7**sendmail.h** Cabecera del fichero sendmail.

7**srvrsmtp.c** Rutinas que implementan el servidor de SMTP.

7**stab.c** Rutinas para el manejo de la tabla de símbolos.

7**stab.c** Rutinas encargadas de manejar y enviar las estadísticas.

7**sysexits.c** Lista de los mensajes de error asociados con los códigos de error en **sysexits.h**.

7**trace.c** El paquete de traza. Estas rutinas comprueban y chequean, haciendo trazas con una alta granularidad.

7**udb.c** El modulo del interface de usuario de la base de datos.

7**usersmtp.c** Rutinas que implementan el usuario SMTP.

7**util.c** Algunas rutinas de propósito general usadas por el sendmail.

7**version.c** Contiene el número de versión y la información acerca de esta versión del sendmail.

Estos son todos los ficheros que posee el sendmail dentro del subdirectorio */src* y los cuales serán compilados para crear el *sendmail*. Como hemos dicho antes para hacer esto teclearemos *sh makesendmail* y este se compilará.

Una vez que hemos compilado el sendmail, lo instalaremos y para ello escribiremos la siguiente línea *sh makesendmail install*. Entonces el fichero binario creado se copiará en *usr/lib* y creará enlaces desde */usr/bin/newaliases* y */usr/bin/mailq* a */usr/sbin/sendmail*.

Ý ya tenemos nuestro fichero binario del sendmail copiado en su lugar, el siguiente paso será crear el fichero de configuración.

3.1. Fichero de configuración

El sendmail no puede trabajar sin un fichero de configuración, el fichero de configuración describe los mecanismos que debe seguir el sendmail, en el lugar donde se encuentra.

El fichero de configuración es bastante ilegible si se intenta ver el código, con lo cual un estudio de lo que es el fichero sería bastante complicado. Para crear el fichero de configuración necesitamos *m4* basado en la configuración de paquetes que una alta complejidad.

Este fichero asume que mas de un equipo utiliza dominios basados en direcciones UUCP; las cuales identifican el nombre del host como "*host!usuario*" en lugar de utilizar "*host.dominio!usuario*". El fichero de configuración puede mejorar esta tarea pero esto puede resultar bastante complejo.

El fichero de configuración será procesado por *m4* para facilitar su creación, en el directorio *cf* encontraremos los ficheros fuentes, dentro de este subdirectorio encontraremos el siguiente conjunto de subdirectorios:

7**cf** Aquí encontramos los ficheros de configuración, con extensión *mc* (Master Configuration), y la salida de estos ficheros serán con extensión cf (Configuration File).

7**domain** Este lugar depende de la descripción de subdominios. Aquí están los ficheros para organizar las direcciones. Por ejemplo **domain/cs.exposed.m4** es la descripción para los host en CS.Berkeley.EDU el subdiminio que encontramos es externo para que el nombre del host sea externamente visible. Estas son referenciados usando la macro DOMAIN **m4** en los ficheros **.mc.**

7**feature** Definiciones y especificaciones de algunos host particulares. Estas referencias se utilizan usando la macro FEATURE **m4**.

7**hack** Local hacks, se referencia usando la macro HACK **m4**.

7**m4** Sitio independiente que incluye toda la información común a los ficheros de configuración. Su referencia está en los *#include*.

7**mailer** Definiciones de los mailer, se referencia utilizando la macro MAILER **M4**. Los tipos de mailer son conocidos cuando las distribuciones son por fax, local, smtp, uucp y usenet.

7**ostype** Definiciones que describen varias directivas de sistemas operativos. Se referencian con la macro OSTYPE **m4.**

7**sh** Ficheros de la shell utilizados para construir procesos con **m4**

7**siteconfig** Información de la conectividad local UUCP. Normalmente contiene listados de los sitios de información

```
SITE(contessa)
SITE(hoptoad)
SITE(nkainc)
```

Estas referencias se hacen con la macro SITECONFIG

```
SITECONFIG(sitio.config.fichero,nom_del_sitio,X)
```

Si se utiliza un nuevo dominio, probablemente se deba de crear un fichero de configuración para el nuevo dominio *cf/domain*. Este consiste principalmente en reglas de definición.

Los subdominios son representados en el directorio *cf/domain*. Por ejemplo el dominio cx-exposed esta en el subdominio de la computadora de redes con el hostname local mostrando otros usuarios; cs-hidden crea usuarios que aparecerán desde el subdominio norba.indexado.com. Si se utilizan subdominios habrá que comprobar el directorio apropiado para el dominio.

El fichero de configuración se creara en *cf/cf* usando un fichero de configuración **.mc**. Dentro de este subdirectorio encontramos el fichero README [Ref3] donde se explica todo con detenimiento.

Para crear el fichero de configuración desde dentro de *cf/cf* debemos de teclear la siguiente línea de comando **m4 ../m4/cf.m4 cf/generic-solaris2.mc >generic.cf**

Donde *generic-solaris2.mc* es el fichero de configuración genérico para solaris, donde incluye solaris 2.5.

3.2. Detalles de los ficheros de instalación

3.2.1. /usr/lib/sendmail

Este es el fichero binario que se crea al compilar el sendmail. Por razones de seguridad este subdirectorio debe de tener el modo 755. También deberían de tener este modo los siguientes directorios */, /usr,* y */usr/sbin*. Esto evitará agujeros de seguridad.

3.2.2. /etc/sendmail.cf

Este es el fichero de configuración para el sendmail. Este y el fichero */etc/sendmail.pid* son los únicos que no se crean al compilar el sendmail.

Estos ficheros se crean como hemos descrito antes.

3.2.3. /usr/bin/newaliases

Los comandos del hoststat deben ser enlazados al *sendmail*:

rm –f /usr/bin/newaliases

ln –s /usr/sbin/sendmail /usr/bin/newaliases

Este fichero puede ser instalado en cualquier otro lugar que se quiera dentro del sistema.

3.2.4. /usr/binhoststat

El comando hoststat debe ser enlazado al *sendmail* de una forma similar al *newaliases*. Este comando lista el estado de las ultimas transacciones del sendmail con otros host remotos.

3.2.5. /usr/bin/purgestat

Este comando debe ser enlazado al sendmail. Contiene toda la información que es almacenada en el arbol del **HoststatusDirectory**.

3.2.6. /var/spool/mqueue

El directorio */var/spool/mqueue* será creado para el mail queue. Este directorio debe de tener el modo 700 y solo puede ser accedido por el root.

3.2.7. /var/spool/mqueue/.hoststat

Este es un tipo calor para las opciones del **HoststatusDirectory**, contiene un fichero por cada host que el sendmail ha utilizado recientemente.

3.2.8. /etc/aliases

El sistema de alises se encuentra en */etc/aliases*. Debe de haber una copia en */lib/aliases* para incluir unas modificaciones en el fichero */etc/aliases*.

cp lib/aliases /etc/aliases

vi /etc/aliases

Los aliases incluidos deben de ser los propios para tu sistema.

Normalmente el sendmail observa este fichero mantenido por rutinas *dbm* o *db*. Otras referencias pueden ser */etc/aliases.dir* y */etc/aliases.pag* dependiendo del paquete de base de datos que se este utilizando. Estos ficheros pueden ser inicialmente creados, pero deben de ser inicializados por el sistema. Deben de tener el modo 644.

```
cp /dev/null /etc/aliases.dir
cp /dev/null /etc/aliases.pag
chmod 644 /etc/aliases.*
newaliases
```

El path actual puede ser modificado en la opción **AliasFile** en el *sendmail.cf.*

3.2.9. /etc/rc

Este fichero es necesario para iniciar el daemon del *sendmail* cuando tu resetees tu sistema. Este fichero realiza dos funciones: Lista los socket SMTP para conexiones (recibir mail desde estaciones remotas) y procesar la cola periódicamente para asegurarse que el correo ha sido liberado cuando el host aceda.

Se deben de añadir las siguientes líneas a */etc/rc* en el área donde comienza el lanzamiento del daemon del sendmail:

```
If [-f /usr/lib/sendmail –a –f /etc/sendmail.cf ]; then
        (cd /var/spool/mqueue; rm –f [lnx]f*)
        /usr/lib/sendmail –fd –q30m &
        echo –n 'sendmail' >/dev/console
fi
```

Los comandos *cd* y *rm* se aseguran que todos los bloqueos de los ficheros han sido quitados; un bloqueo extraño puede hacer que en la mitad de la recepción de un mensaje en proceso se venga abajo. La línea que invoca al *sendmail* con dos flags "*–bd*" causa un listado en el puerto SMTP y "*–q30m*" causa la ejecución de la cola cada media hora.

Si no estás ejecutando una versión de UNIX que soporte Berkeley TCP/IP no se debe incluir el flag *–bd*.

3.2.10. /usr/lib/sendmail.hf

Este es el fichero de ayuda utilizado por los comandos de ayuda del SMTP.

El path actual de este fichero está definido en la opción H del *sendmail.cf.*

3.2.11. /etc/sendmail.st

Aquí encontramos una colección estadística del trafico de mail. Para crear este fichero:

```
cp /dev/null /etc/sendmail.st
chmod 666 /etc/sendmail.st
```

Este fichero se imprime con el programa "*mailstats/mailstats.c*". El path actual de este fichero está definido en la opción **S** del fichero *sendmail.cf*.

3.2.12. /usr/mailq

Si el sendmail es invocado como *mailq*, se simulará el flag **–bp** (esto imprimirá el contenido de la cola de mensajes). Este debe tener un enlace a */usr/lib/sendmail*.

3.3. La cruda realidad

Ya sabemos como crear el fichero binario del sendmail, sabemos también como crear el fichero de configuración, tenemos información sobre todos los ficheros que tenemos, como compilar, ejecutar o modificar cada uno de ellos, pues llegó el momento de pasar a la acción y compilar el sendmail.

Todo el proceso que se llevó a cabo para la instalación, está explicado en las memorias que se adjuntan, aquí vamos a hacer un breve repaso de cual es el proceso y posibles errores que se pueden producir, y cuales hemos detectado nosotros.

La primera advertencia que se nos hace va referida al compilador GCC sobre todo a la versión 2.5.x, por lo que se indica en el fichero de instalación [Ref1], tiene problemas con las definiciones de algunas constantes, siendo imposible compilar el sendmail, por lo cual se recomienda no utilizar esta versión.

Otro problema con el GCC 2.7.x . Se dice que en los procesadores Pentium existen problemas de optimización. Y se recomienda utilizar el flag –O en esta arquitectura.

Problemas con GDBM. El problema está en que sendmail 8.8 no trabaja con GDBM por causas de seguridad.

Existe un problema que nos ocurrió a nosotros con la librería *l44bsd* ya que no aparece en todos los sistemas, pero eliminando este parámetro de las librerías, no se produce ningún problema. Esta línea se encuentra en *makefile*.

3.3.1. Problemas específicos de Solaris 2.x (SunOS 5.x)

Si compilamos en Solaris, el fichero *makesendmail* debe de ser modificado para indicar la versión de Solaris incluyendo una definición (ej. –DSOLARIS=2400 para 2.4 o –DSOLARIS=20501 para 2.5.1). Si se utiliza **gcc**, asegúrese de no utilizar *–I /usr/include*. Si se utiliza **cc** de Sun utilice */opt/SUNWspro/bin/cc* usando la instancia de */usr/ucb/cc*.

En principio no existen mas problemas para esta arquitectura, ya que los problemas que se nos produjeron a nosotros no son fallos del sendmail.

Si ya hemos ejecutado la línea *sh makesendmail* y *sh makesendmail install* ya tenemos nuestro fichero ejecutable en su lugar y compilado, ahora creamos el fichero de configuración como se explico anteriormente, y no debe de dar ningún problema, una vez creado como **generic.cf** lo copiamos con el nombre **sendmail.cf** a su lugar */etc*. Y ya podemos lanzar nuestro daemon del sendmail. Miramos la cadena de arranque del sendmail, y ejecutamos **./k57sendmail start**. Y el sendmail queda instalado y funcionando.

3.3.2. Comprobando que funciona

Ya lo tenemos instalado y funcionando pero eso debemos de probarlo, lo primero que hacemos es un telnet a nuestro puesto en el puerto SMTP y nos debe de responder el sendmail indicando la versión que se está ejecutando.

telnet norba smtp

La respuesta es satisfactoria.

 Primer mensaje:

Este mensaje se envió y se recibió correctamente desde la cuenta de norba.

```
From - Tue May 19 15:22:05 1998
Received: from cybercursos.net (64.66.177.193) by cybercursos.net
          with ESMTP (Apple Internet Mail Server 1.1); Fri, 15 May 1998 19:47:54
+0200
Sender: gabriel@cybercursos.net
Message-ID: <355C9B60.324F78E1@cybercursos.net>
Date: Fri, 15 May 1998 19:45:37 +0000
From: Webmaster Cybercursos <gabriel@cybercursos.net>
Reply-To: gabriel@cybercursos.net
X-Mailer: Mozilla 4.04 [en] (X11; I; Linux 2.0.33 i586)
MIME-Version: 1.0
To: comentarios@cybercursos.net
Subject: probando sendmail 1
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Status: R
X-Mozilla-Status: 0001
Content-Length: 45
        Primer mensaje enviado al sendmail de comentarios
```

 Segunda prueba.

Esta prueba consistía en probar el fichero de configuración de host, este fichero es el *sendmail.cw*, este fichero se supone que debe de llevar los nombres de los host que

```
From - Tue May 19 15:22:05 1998
Received: from cybercursos.net (64.66.177.193) by cybercursos.net
with ESMTP (Apple Internet Mail Server 1.1); Fri, 15 May 1998 19:51:46 +0200
Sender: gabriel@cybercursos.net
Message-ID: <355C9C49.53722DA@cybercursos.net>
Date: Fri, 15 May 1998 19:49:29 +0000
From: Webmaster Cybercursos <gabriel@cybercursos.net>
Reply-To: gabriel@cybercursos.net
X-Mailer: Mozilla 4.04 [en] (X11; I; Linux 2.0.33 i586)
MIME-Version: 1.0
To: gabriel@cybercursos.net
Subject: prueba 2 del sendmail,
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Status: R
X-Mozilla-Status: 0001
Content-Length: 59

        esta prueba debe de fallar al no tener aceptado este host
```

pueden aceptar correo, pero este fichero no funciona, o por lo menos no hace su cometido, se envió un mensaje a **sande** que es uno de los nombres que tiene norba y el cual no estaba incluido en el fichero *sendmail.cw*, pero este recibió el mensaje perfectamente.

Con estas dos pruebas podemos asegurar que el sendmail funciona perfectamente. Ahora instalaremos el daemon del protocolo POP, siguiendo los mismos pasos que para el sendmail. Es decir que primero haremos una introducción a este protocolo para después hablar del programa utilizado y como instalarlo, así como las pruebas realizadas.

4. Introducción al protocolo POP

El protocolo POP es el encargado de enviar y recoger los mensajes entre el servidor y nuestro puesto de trabajo. Este permite que una estación de trabajo aceda dinámicamente a nuestro buzón en el servidor host. Para explicar este protocolo nos referimos al "*cliente host*" como la máquina host que utiliza el servicio POP3, mientras que el termino "*servidor host*" se refiere al host que ofrece el servicio POP3.

Cuando un usuario agente o un cliente host introducen un mensaje en el sistema de transporte este establece una conexión SMTP al host relativo.

Inicialmente, el servidor host comienza el servicio POP3 a través del puerto TCP 110. Cuando un cliente host llama a un usuario el servicio establece una conexión TCP con el servidor host. Cuando la conexión es establecida, el servidor de POP3 envía una contestación de aceptación. El cliente y el servidor POP3 intercambian comandos y respuestas hasta que la conexión es cerrada o abortada.

Los comandos en POP3 consisten en unas posibles teclas seguidas de un argumento. Todos los comandos terminarán por un par de caracteres CRLF.

La respuesta del POP3 consiste en indicar un suceso, y responder con la posible información. Todas las respuestas son terminadas por un par de caracteres CRLF. Existen dos formas de confirmar un proceso: positivamente ("+OK") y negativamente ("-OK").

La respuesta a ciertos comandos puede ser multi-línea. En estos casos, cuando son borrados los identificadores, después de enviar la primera línea de la respuesta y un CRLF, una línea adicional es enviada, con la terminación de dos CRLF. Cuando todas las líneas enviadas han obtenido su respuesta, se envía un final de línea, que consiste en un octeto de terminación (código decimal 046, ".") y un par de CRLF. Si una línea de una respuesta multilínea responde comenzando con el octeto de terminación, la línea es "byte-stuffed" por el pre-envío del octeto de terminación que había en la línea de respuesta. La respuesta en una multilínea se puede terminar con cinco octetos "CRLF.CRLF". Cuando examinamos una multilínea de respuesta, el cliente comprueba si el comienzo de la línea contiene los octetos de terminación. Si aparecen otros caracteres después de la secuencia de CRLF, el primer octeto de la línea es reenviado de nuevo. Si solo es un CRLF inmediatamente envía el carácter ce terminación, entonces la respuesta del servidor de POP es terminar el contenido de la línea con "*.CRLF*" y no se considera parte de la multilínea.

Una sesión de POP3 progresa a través de un número de estados durante su tiempo de duración. Una vez que la conexión TCP ha sido abierta y el servidor de POP3 ha enviado la gratificación, la sesión entra en un estado de **autorización** En este estado el cliente debe de identificarse a si mismo al servidor de POP3. Uno de los clientes entonces adquiere los recursos asociados al servidor con el buzón del cliente, y la sesión entra en el estado de **transacción** En este estado, el cliente requiere aciones de parte del servidor de POP3. Cuando el cliente ha finalizado la transacción, la sesión entra en el estado de **actualización**. En este estado el servidor de POP3 revisa los recursos adquiridos durante el estado de transacción y se despide. La conexión TCP es entonces cerrada.

4.1. El estado de autorización

Una de las conexiones ha sido abierta por el cliente POP3, el servidor POP3 envía una línea de gratificación. Esta será una cadena terminada por CRLF. Un ejemplo sería:

S. +OK dewey POP3 server ready (comments to: jj@indexado.com)

Esta será la respuesta de gratificación del POP3. El servidor siempre tiene una respuesta positiva para la gratificación.

La sesión de POP3 está ahora en el estado de **autorización** El cliente debe de utilizar los comandos de usuario. Si el servidor de POP3 responde con un suceso positivo, indicador "*+OK*" entonces el cliente debe de pasar el comando completo de autorización, o enviar un comando de salida para terminar la sesión POP3. Si el servidor de POP3 responde con un suceso negativo "*-ERR*" al comando del usuario, entonces el cliente debe de introducir un nuevo comando de usuario, o enviar el comando de finalizar.

Cuando el cliente utiliza los comandos pasados, el servidor de POP3 utiliza los argumentos pasados desde el usuario y los comandos determinarán si el cliente puede acceder correctamente a su buzón. También ocurrirá así si el servidor de POP3 adquiere un acceso de bloqueo exclusivo al buzón. Si el bloqueo es adquirido, el servidor de POP3 pasará los mensajes individuales al buzón, determinado cual fue el último mensaje presente en el buzón que hacía referencia por el comando RETR y fue respuesto con un suceso positivo. La sesión de POP3 entrará ahora en el estado de **transacción**. Si el bloqueo no puede ser adquirido o el cliente deniega el acceso al buzón o el buzón no puede ser leído por alguna razón, el servidor de POP3 responde con un indicador de suceso negativo.


La mínima implementación del POP3 necesita solo acceder al buzón y adquirir todos los mensajes que lo componen, no tiene porque hacerlo para mensajes individuales. En implementaciones mas avanzadas si debe tener esta capacidad.

Después de que el servidor de POP3 halla recuperado los mensajes individuales del buzón, este les asigna un id-mensaje a cada mensaje, y anota el tamaño de cada mensaje en octetos. El primer mensaje en el buzón es asignado con el id-mensaje "1", el segundo es asignado "2" y así sucesivamente hasta "n". En los comandos del POP3 y respuestas, todos los id-mensajes y tamaños, son expresados en base-10 (decimal),

Estos serían los tres comandos POP3 para este estado.

USER name

Argumento: el servidor especifica id-usuario.

Restricciones: Solo pueden indicarse en el estado de autorización después de la confirmación del POP3 o después de un comando del usuario.

Posibles respuestas:

+OK name is welcome here

-ERR never heard of name

Ejemplo:

C: USER mrose

S: +OK mrose is a real hoopy frood

……

C: USER frated

S: -ERR sorry, frated doesn't get his mail here

PASS string

Argumento: debe de especificarse un password para el id-usuario.
Restricciones: Solo puede ocurrir en el estado de autorización , después del comando USER.
Posibles respuestas:
+OK maildrop locked and ready
-ERR invalid password
-ERR unable to lock maildrop
Ejemplos:
C: USER mrose
S: +OK mrose is a real hoopy frood
C: PASS secret
S: +OK mrose's maildrop has 2 messages
(320 octetos)
.....
C: USER mrose
S: +OK mrose is a real hoopy frood
C: PASS secret
S: -ERR mrose's to lock mrose's maildrop ,file already locked.

QUIT

Argumentos: ninguno
Restricciones: ninguna
Posibles respuestas:
+OK
Ejemplos:
C: QUIT
S: +OK dewey POP3 server signing off

4.2. El estado de transacción

Uno de los clientes ha sido identificado por el servidor de POP3 y el servidor de POP3, tiene bloqueado su buzón , la sesión de POP3 está ahora en el estado de **transacción**. El cliente debe ahora introducir los comandos del POP3 respectivos a esta sesión. Después de cada comando el servidor recibirá una respuesta. Eventualmente el cliente utilizará el comando QUIT y la sesión de POP3 entrará en estado de actualización.

Vamos a ver ahora los comandos de que disponemos en el estado de transacción:

STAT

Argumentos: ninguno
Restricciones: solo puede ejecutarse en el estado de transacción.

Explicación: El servidor de POP3 envía una respuesta positiva cuando una línea contiene información para el buzón. Esta línea es llamada una "lista de correo" para ese buzón.
Todos los servidores de POP3 requieren un cierto formato para esta lista de correo. El primer octeto debe indicar el número de mensajes en el buzón. Lo siguiente es el tamaño del buzón en octetos.


Esta es la mínima información que se debe facilitar pero en función de la implementación del POP3 se facilitará más información. Además los mensajes borrados no se tienen en cuenta.

Posibles respuestas:
+OK nn mm
Ejemplos:
C: STAT
S: +OK 2 320

LIST [msg]

Argumentos: un identificador de mensaje. Este identificador no puede referirse a un mensaje marcado como borrado.
Restricciones: solo puede utilizarse en el estado de transacción.
Explicación: Si el argumento es aceptado por el POP3, entonces pasará a ser una multilínea. Después del +OK, para cada mensaje en el buzón, el servidor de POP3 responderá con una línea que contiene información para cada mensaje. Esta línea es llamada "lista de datos" para cada mensaje.
La forma de esta línea viene definida, primero por un octeto con el identificador, después el tamaño en octetos, y esto será para la mínima implementación. Para implementaciones mas avanzadas se incluirá más información.


Los mensajes marcados como borrados no se incluirán en el listado.

Posibles respuestas:
+OK scan listing follows
-ERR no such message
Ejemplos:
C: LIST
S: +OK 2 messages (320 octetos)
S: 1 120

S: 2 200
S: .
.....
C: LIST 2
S: +OK 2 200
.....
C: LIST 3
S: -ERR no such message, only 2 messages in maildrop

RETR msg

Argumentos: un id-mensaje (obligatorio). Este id-mensaje no debe referirse a un mensaje marcado como borrado.
Restricciones: solo en el estado de transacción.
Explicación: Si el servidor de POP3 responde positivamente, entonces se pasará a multilínea. Después del +OK el servidor de POP3 enviará el mensaje correspondiente al id-mensaje.
Si el número asociado con este mensaje es mayor que el número de accesos mas alto en el buzón, el servidor de POP3 actualizará el número de accesos más alto al número asociado con este mensaje.
Posibles respuestas:
+OK message follows
-ERR no such message
Ejemplos:
C: RETR 1
S: +OK 120 octetos
S: <el servidor de POP3 envía el mensaje entero aquí>
S: .

DELE msg

Argumentos: un id-mensaje. Este id-mensaje no puede referirse a un mensaje marcado como borrado.
Restricciones: solo en el estado de transacción
Explicación: El servidor de POP3 marca el mensaje como borrado. Una futura referencia al id-mensaje asociado con un comando del POP3 generará un error. Este mensaje no será borrado hasta que la sesión de POP3 entre en estado de actualización.
Si el número asociado con este mensaje es mayor que el número de acceso mas alto en el buzón, el servidor de POP3 actualizará el número de accesos más alto con el número asociado a este mensaje.

Posibles respuestas:
+OK message deleted
-ERR no such message
Ejemplos:
C: DELE 1
S: +OK message 1 deleted
.....
C: DELE 2
S: -ERR message 2 already deleted

NOOP

Argumentos: ninguno
Restricciones: solo en el estado de transacción
Explicación: El servidor de POP3 no hace nada, solo puede esperar una respuesta positiva.
Posibles respuestas:
+OK
Ejemplos:
C: NOOP
S: +OK

LAST

Argumentos: ninguno
Restricciones: solo en el estado de transacción
Explicación: El servidor POP3 utiliza una respuesta positiva cuando una línea contiene el mayor número de mensaje accedido. Se devuelve cero en caso de no haber mensajes en el buzón. Un cliente puede transferir ese mensaje, si existe un número mayor que el respondido en el comando LAST estos mensajes no serán accedidos por el cliente.
Posible respuesta:
+OK nn
Ejemplos:
C: STAT
S: +OK 4 320
C: LAST
S: +OK 1
C: RETR 3
S: +OK 120 octets
S: <el servidor de POP3 envía el mensaje entero>
S: .
C: LAST
S: +OK 3
C: DELE 2
S: +OK message 2 deleted

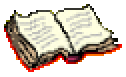
C: LAST
S: +OK 3
C: RSET
S: +OK
C: LAST
S: +OK 1

RSET

Argumentos: ninguno
Restricciones: solo en el estado de transacción
Explicación: Si un mensaje ha sido marcado como borrado por el servidor POP3, este será desmarcado. El servidor POP3 responderá entonces con una respuesta positiva. El valor del número de acceso mas alto será restablecido a su valor inicial.
Posibles respuestas:
+OK
Ejemplos:
C: RSET
S: +OK maildrop has 2 messages (320 octets)

4.3. El estado de actualización

Cuando el cliente utiliza el comando QUIT desde el estado de transacción, la sesión de POP3 entrará en el estado de actualización.



Si el cliente utiliza el comando QUIT desde el estado de autorización, la sesión de POP3 terminará sin pasar por el estado de actualización.

QUIT

Argumentos: ninguno
Restricciones: ninguna
Explicación: El servidor borrará todos los mensajes marcados para borrar desde el buzón. Si la revisión está en bloqueo de acceso exclusivo el buzón responderá a este suceso con las operaciones realizadas. La conexión TCP será finalizada.
Posibles respuestas:
+OK
Ejemplos:
C: QUIT
S: +OK dewey POP3 server signing off (maildrop empty)
.....
C: QUIT
S: +OK dewey POP3 server signing off (2 messages left)

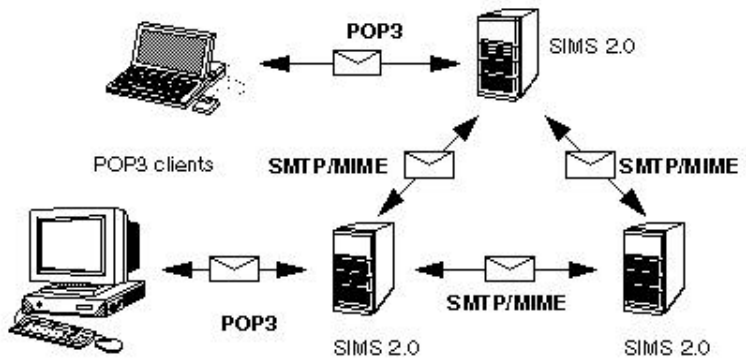
.....

No debemos de confundir el POP como protocolo con el daemon de POP, lo que acabamos de ver es el protocolo POP3 y sus comandos, para ver el funcionamiento de estos comandos o trabajar directamente con el Servidor POP3 debemos de hacer un telnet al puerto 110 donde nos responderá el Servidor de POP3 y podremos introducir los comandos vistos. Los comandos para el Qpopper o daemon de POP los veremos a continuación.

telnet norba 110

Los comandos que hemos visto son para una implementación mínima del protocolo POP3, existen otros comandos para otros tipos de implementaciones del POP3, pero los cuales no vamos a ver, para más información referirse al RFC 1081 y 1082.

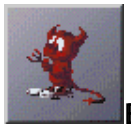
La figura que aquí se ve representa el funcionamiento del servicio de correo, indicando cual es la función de los protocolos POP3 y SMTP.



5. Pasos previos a la instalación

Ya hemos visto como funciona el protocolo POP, ahora vamos a centrarnos en nuestro daemon, que características posee, como compilarlo para su mejor aprovechamiento, y como lanzarlo para que funcione correctamente.

El programa que nosotros vamos a utilizar es el **Qpopper** en su versión **2.4**, este programa actúa como servidor de POP, este programa ha sido probado en varios máquinas UNIX, funcionando perfectamente, aunque los proveedores no se hacen responsables de cualquier fallo que pueda producir, ni aseguran su perfecto funcionamiento, sin embargo este servidor ha sido instalado y está funcionando en varios servidores de correo sin ningún problema.



El Qpopper 2.4

Este servidor de POP pertenece a Berkeley, y es un simple programa que es lanzado por *inetd* (fichero de arranque), obteniendo este respuestas en el puerto TCP POP.

El puerto por defecto es el 110, aunque algunos clientes utilizan el 109

El programa Popper inicializa y verifica que las direcciones IP están registradas en el dominio local, enviando un mensaje de error cuando una conexión hecha por un cliente cuya dirección IP no tiene asignado un nombre. En los sistemas que utilizan BSD 4.3, este también chequeará si los nombres canónicos están bloqueados para el cliente retornando a la dirección IP un mensaje de error, indicando el fallo de acceso. El servidor tiene un estado de autorización, durante el cual el cliente que aceda a el debe de identificarse con su login y password correspondiente para el servidor. No se permiten otro tipo de cambios durante la ejecución (exceptuando la salida). Si la autentificación fallara, se enviará un mensaje de error, y la sesión finalizará. Una vez que el usuario ha sido identificado, Qpopper cambia al estado de transacción entrando en su grupo de trabajo. El servidor crea una copia del mail del usuario (generalmente en */usr/spool/mail*), el cual es utilizado para todas las secuencias de las transacciones. Esto incluye los comandos del POP, respuesta de correo, borrado de correo, recuperación de correo, y el resto de comandos disponibles. Una extensión será lanzada por el usuario a una parcela de mail a el servidor que está utilizando sendmail (esta extensión es soportada por el cliente HyperMail distribuido con el servidor). Cuando la conexión a la red ha terminado ser recupera la copia realizada actualizando las posibles modificaciones sobre el buzón temporal.

El servidor de POP utiliza *syslog* para registrar sus actividades. En un sistema con **syslogging** BSD 4.3, el servidor entra en el local0 asignando la prioridad "*notice*" para todos los mensajes exceptuando los debugging los cuales tienen prioridad "*debug*". Por defecto el fichero de log está en */usr/spool/mqueue/POPlog*. Este puede ser cambiado si se desea. En el sistema con **syslogging** 4.2 todos los mensajes son entrados en el fichero local de log, usualmente */usr/spool/mqueu/syslog*.

5.1. Problemas

Si el fichero de sistema de recepción del mail se encuentra lleno en */usr/spool/mail* los usuarios tendrán problemas. El fichero de sistema debe tener espacio suficiente para dos copias de el mensaje mas largo que pueda llegar. Popper tiene un diseño robusto en lo que respecta a este problema, pero si te encuentras con esta situación debes cerrar la sesión, y tus ficheros se encontrarán en

/usr/spool/mail/.userid.pop

y el resto de correo se encontrará en

/usr/spool/mail/userid

Si esto ocurre el administrador debe de liberar espacio en disco para que los ficheros del sistema dispongan de espacio libre en los buzones de correo. Entonces el usuario cuando inicie una sesión de POP u otro programa de mail debe de borrar sus mensajes de correo.

Una alternativa sería que el administrador del sistema, combinara estos dos ficheros, pero esta solución no es muy recomendable a no ser que no se disponga de suficiente espacio libre.


En el manual [Ref4] del Qpopper no se hacen referencias a ningún problema más, luego pasaremos al siguiente punto.

5.2. Limitaciones

El servidor de POP copia la carpeta entera de correo del usuario al subdirectorio temporal y trabaja con esa copia. Si el buzón es excesivamente grande o no hay espacio suficiente en el subdirectorio temporal */tmp*, entonces el servidor se negará a continuar y terminará la conexión.

Las modificaciones simultaneas de la misma carpeta de correo pueda producir resultados erróneos. Por ejemplo la manipulación de un mensaje usando el comando **mail** en UNIX en */usr/ucb/mail* mientras este está siendo procesado por el servidor de POP puede causar que los cambios hechos por una de los programas sean perdidos cuando el otro programa termine. Este problema se está estudiando y espera ser arreglado en versiones posteriores.

5.3. Otras características


Si añadimos el parámetro –s a la línea del comando se generará una estadística en el log. Este se efectuará para cada sesión de cada usuario.

Ejemplo:

```
Stats: nombreusuario aaa  bbb ccc ddd

Donde:
aaa= número de mensajes borrados
bbb= número de bytes borrados
ccc= número de mensajes en el servidor
ddd= número de bytes en el servidor
```


Se puede sumar un boletín de POP. Esta característica sirve al administrador del sistema para enviar importantes anuncios a todos los usuarios POP que tenga incluidos en la lista de mailing.

Esta característica se activa con el flag –b en la línea de comandos. Esta opción posee un directorio de boletines.

El directorio de boletines posee un fichero por cada boletín. Cada fichero contiene un mensaje completo con cabecera y cuerpo en el formato normal. La primera línea de cada boletín debe tener un "**From:**". La forma de enviar los boletines para el administrador del sistema es enviando a sí mismo una copia del boletín (usando una cuenta donde se encuentran las direcciones a las que enviar el correo), entonces usando el programa de mail se salva el mensaje a un fichero en directorio de boletines en el formato normal. El subdirectorio de boletines debe de ser accesible por todo el mundo.

El nombre de cada boletín comenzará con un número de boletín, y opcionalmente continuará con otro carácter.

Popper creará un fichero llamado **.popbull** en el directorio home de cada usuario. Este fichero contiene una simple línea recordando el último número de boletín recibido por el usuario.

Los boletines son procesados por el Popper en *pop_dropcopy.c* inmediatamente después de copiar el correo a la carpeta temporal, se construirá un índice a la carpeta temporal. Todos los boletines que este usuario no ha recibido previamente serán abiertos en un fichero temporal en su buzón.

Cuando los boletines son copiados al buzón temporal, todas la cabeceras "**To:**" serán reemplazadas por "**To: nombreusuario@myhost**", y la línea **Status** será borrada. En otro caso los boletines serán copiados normalmente.

Cuando un nuevo usuario comprueba por primera vez el correo, popper creará el fichero **.popull** en el directorio home del usuario y recibirá el último boletín. Este nuevo usuario no obtendrá los boletines anteriores.

 Se puede cambiar el tiempo de salida desde 30 a 300 segundos (5 minutos). Este valor puede ser razonable, para algunas conexiones lentas.

6. Comenzando la instalación

Ya se ha explicado cuales son las características y posibilidades del Qpopper, ahora vamos a explicar como instalarlo, primero veremos de que ficheros disponemos para la instalación del Qpopper, seguidamente explicaremos como compilar estos ficheros, explicaremos también cuales son los parámetros de entrada que puede recibir el Qpopper, y acabaremos con la parte práctica.

6.1. Detalles de los ficheros del código fuente

7**version.h** Contiene la versión del Qpopper.

7**popper.c** Contiene las llamadas y señales para establecer las conexiones. Popper tiene una señal SIGPIPE si el cliente aborta la cadena TCP mientras recibe datos. Popper debe reescribir el fichero del buzón en este caso.

7**pop_pass.c** Suma un modo desde Don Lewis <gdonl@gv.ssil.com>. Bajo SunOs 4.1.3, y posiblemente otros sistemas, el chequeo de passwords nulos con los cuales no trabaja. QC3 chequea solo para un puntero a un password nulo en la estructura retornada por getpwnam.

7**pop_updt.c** Suma los códigos de entrada stats. Si el flag –s es especificado en la línea de comandos, el siguiente mensaje es generado:

```
Stats: nombreusuario aaa bbb ccc ddd
Donde:
aaa= número de mensajes borrados
bbb= número de bytes borrados
ccc= número de mensajes en el servidor
ddd= número de bytes en el servidor
```

7**popper.h** Suma un nuevo campo llamado "*stats*" a la estructura del POP. Este campo es un entero no nulo si el stats es requerido.

Suma un nuevo campo llamado "*bulldir*" a la estructura del POP. Este campo almacena el path del directorio de boletines, o es nulo si la opción –b no está activada.

Incluye una declaración externa para pop_xlst.

Se puede cambiar el tiempo de salida de 30 a 300 segundos. Por defecto el valor es de 30 segundos.

7**pop_init.c** Procesa el parámetro –s de la línea de comandos.

Procesa el parámetro –b de la línea de comandos.

7**pop_bull.c** Nuevo fichero fuente para los boletines del sistema.

7**pop_dropcopy.c** Añade las llamadas a pop_bull

7**xtnd_xlst.c** Fichero fuente para los comandos **xlst**.

7**pop_get_subcommand.c** Suma una tabla de entrada para los comandos **xlst**.

7**Makefile** Suma *pop_bull.c* y *xtnd_xlst.c* al código fuente creando los ficheros obj.

7**popper.8** Actualización de las páginas del man donde se describen los cambios.

6.2. Compilando el Qpopper

La fase de compilación del Qpopper resulta fácil y no suele dar problemas, los pasos que debemos de seguir son cuatro:

1. Compilamos el código fuente para crear el fichero binario, para ello desde el directorio donde se encuentre el código fuente tecleamos los siguientes comandos:

```
./configure
make
```

Ya tenemos nuestros ficheros ejecutables **popper** y **popauth**.

2. copiar los ficheros ejecutables al directorio que nosotros deseemos que se encuentren, generalmente a */usr/sbin* o */usr/lib*.
3. Debemos de modificar el fichero de **inetd.conf** para añadir una nueva cadena de arranque que soporte esta versión del Qpopper.

```
Pop3 stream tcp nowait root /usr/sbin/popper popper –s
```

Esta es la línea que debemos de añadir.

Si nuestro sistema operativo no tiene el fichero inetd.conf debe usarse el fichero de configuración /etc/servers, añadiendo la siguiente línea

```
Pop3 tcp /usr/sbin/qpopper qpopper –s
```

4. Por último debemos de modificar el fichero de servicios */etc/services* necesitando incluir la siguiente línea.

```
Pop3        110/tcp
```

Una vez ejecutados estos pasos, para concluir debemos de reiniciar el fichero inetd, para ello ejecutamos Kill –HUP <inetpid>. Lo normal es que se reinicie automáticamente, de no ser así ejecutaremos:

```
inet imp
refresh –s inetd
```

y con esto queda instalado y compilado el Qpopper.

6.2.1. Posibles errores que se pueden producir después

Para probar si funciona haremos un **telnet** al puerto del qpopper, teclearemos "*telnet <qpopper host> Pop3*" si el Qpopper responde correctamente quiere decir que funciona, nos pedirá nuestro nombre y clave, de tener algún fallo nos dará dos posibles mensajes:

1. "connect: Connectio refused"
2. "connect: Connection closed"

Si se recibe el mensaje 1, comprueba tu fichero de servicios y asegúrate de haber creado el puerto correspondiente al "POP3" y de que el nombre asignado es el mismo que has puesto en el fichero *inetd.conf*. Esto también puede indicar que el *inetd* no ha sido reiniciado, luego ejecute *kill –HUP <inet pid>*.

Si se recibe el mensaje 2, esto indicará que inetd no tiene asignado el puerto correcto al qpopper, por eso el programa no puedo localizarlo, o falla en la conexión. Revise el fichero *inetd.conf*, también puede sumar el flag –d y chequeará sus mensajes largos para saber la fuente del problema.

Si la instalación fue satisfactoria recibirá el siguiente mensaje:

+OK QPOP (version 2.4) at <system> starting. <13625.811191280@system>
introduzca su nombre como
user <nombre usuario>
+OK Password required for <nombre usuario>
pass <el password>
+OK mark has 2 message(s) (4123 octets).
Y pulsando el número leerá los mensajes.

6.3. Parámetros del Qpopper

7-**b bulldir** Siendo bulldir la localización del directorio de boletines. Este comando sobreescribe los valores compilados que habían sido definidos. Lee los nombres de los ficheros que pueden ser creados usando la forma **número.cadena**. Por ejemplo 00001.Bulletin_one, 0002.2hr_Downtime_2-4-98. Esta característica puede ser activada en la compilación activado –DBULLDB, requiere dos ficheros en blanco incluyendo el nombre de BULLDIR/bulldb.pag y la creación de BULLDIR/bulldb.dir.

7-**d** Activa el login del debug si compila.

7-**k** Si compila con –DKERBEROS, esta flag activa solo el soporte **kerberos**. En qpopper introduzca este comando en el puerto apropiado **kpop** en **inetd**.

7-**s** Este activa la entrada de estadísticas.

7-**t logfile** Si son definidos los ficheros de traza y debug, la salida desde el login escribirá una traza en el fichero de log.

7-**T timeout** Sirve para poder cambiar el tiempo por defecto para leer el tiempo de salida de mensajes. Esto causa que el qpopper cierre la conexión cuando pase este tiempo. Varía entre 30 y 120 segundos.

6.4. La cruda realidad

Ya sabemos como compilar el **qpopper**, también sabemos como adaptarlo a nuestras necesidades como modificarlo, así como todos los ficheros que posee. Y sabemos como compilarlo y ejecutarlo.

Seguimos los pasos que se indicaron anteriormente. Lo compilamos lo copiamos a su lugar, introducimos la cadena de arranque, y lanzamos el qpopper, y todo esto se hace sin ningún problema aunque parezca milagroso. En la memoria que se acompaña, se explica como se hizo. En caso de que se produzca algún error, ya hemos explicado también como solucionar estos posibles problemas.

Por último vamos a crear una base de datos para claves encriptadas, esto se utiliza para métodos de autentificación. Para activar esta característica debemos de definir las siguientes sentencias de compilació n:

```
APOP=\"/etc/pop.auth\"
POPUID=\"pop\"
```

La primera definición es la localización de la base de datos; la segunda especificación es la entrada que tomará el usuario para la autorización de la clave.

Cuando se construye el **qpopper** con APOP, se obtendrá un programa llamado **popauth** el cual debe ser instalado en una localización pública. Este programa de ejecutar SUID como el usuario de pop el cual no puede hacer modificaciones en la base de datos **pop.auth**.

Asegúrese de que la base de datos /etc/pop.auth tiene asignado POPUID y que el permiso que posee es 600. Popauth –init creará el fichero con el propio owner y permisos.

La base de datos debe ser inicializada por el root, y teclearemos el siguiente comando:

```
popauth – init
```

para sumar nuevos usuarios teclearemos

```
popauth – user <user>
```

o los borraremos con

```
popauth – delete <user>
```

para añadir los cambios ejecutaremos

```
popauth
```

y ya tenemos instalado nuestro cliente de POP.

6.4.1. Comprobando que funciona

Para probarlo enviamos un mensaje desde norba hacia fuera y vemos que efectivamente funciona como debería.

REFERENCIAS:

[Ref1]: Fichero perteneciente al Sendmail 8.8.8, este fichero se encuentra dentro del subdirectorio */Sendmail-8.8.8/*READ_ME, y posee información para la instalación, así como posibles problemas que pueden surgir. También se explica cuales son las variables de entorno que utiliza el sendmail.

[Ref2]: Este fichero se encuentra dentro de la documentación que acompaña al Sendmail 8.8.8, está en formato PostScript para su impresión desde cualquier programa que lea este tipo de ficheros, esta guía de instalación es bastante completa, está editada por Eric Allman (eric@Sendmail.ORG) y esta es su versión 8.106 y sirve para el Sendmail 8.8, en ella se habla del proceso de instalación, compilación, se describen todos los ficheros fuentes y de configuración, los comandos que acepta el sendmail, los parámetros que se pueden introducir en la línea de comandos. Y se habla de cómo modificar los ficheros de configuración y fuentes del Sendmail, así como los posibles errores que se conocen y como solucionarlos.

[Ref3]: Este fichero se refiere a todo lo que es el proceso de configuración del sendmail, fichero **sendmail.cf**, se cuenta como crear este fichero, cual es su utilidad, posibles errores de creación, y las variables de entorno que tiene, para que sirven y como se podrían modificar.

[Ref4]: Este fichero corresponde al Qpopper, dentro de su directorio, encontramos un fichero README.Berkeley, donde encontramos toda la información referente a la compilación, instalación, errores, modificaciones y comandos del Qpopper.