Revista para nuevos usuarios de GNU/Linux

sólo LINUX

Número 15 • 995 ptas. • 5,98€ • 990 ESC (CONT) • Segunda época
www.prensatecnica.com

Prens@ Técnic
8 413042 087359 00015

Entrevista
El Director General de SCO para España y Portugal, nos habla de la compra de su empresa por Caldera Systems

Samba
Heramientas gráficas para trabajar con Samba

LaTex
Descubre sus clases y estilos

Primeros Pasos
Procesado de correo electrónico con procmail

Espiral

Una comunidad de Debian

2 CD-Roms

CD1:
- **Red Hat 7.0** (instalación)
  - Distribución básica
  - Amplia documentación
  - Entornos de ventana

CD2:
- **Red Hat 7.0** (programas)
  - RPM 4.0
  - Xfree 4.0.1
  - Gnome 1.2
  - Kernel 2.2.16
  - OpenSSH

Santiago Romero

Procesado de correo con procmail

Una vez disponemos del correo en nuestra máquina necesitamos clasificarlo. Esto nos permitirá separar el correo en diferentes carpetas, leer con más comodidad las listas de correo, borrar mensajes no deseados, tener autorespuestas, etc. Todo esto es posible gracias a PROCMAIL.

Muchas veces nos gustaría distribuir el correo que nos llega en diferentes ficheros o cuentas, de forma que puedan leerse separadamente los e-mails personales de los cientos de mensajes de las listas de correo, mientras que en otras ocasiones nos gustaría que hubiese una manera de auto responder ciertos e-mails o de eliminar correo de propaganda o spam.

Procmail es un procesador de correo autónomo que se ejecutará en nuestra máquina al producirse la llegada de un nuevo mensaje. En ese momento procmail será llamado automáticamente gracias al fichero *.forward* (como veremos en este artículo) o a través de sendmail, de tal modo que leerá el correo de la entrada estándar y abrirá el fichero *.procmailrc* que le indicará qué debe hacer con ese correo en función de una serie de reglas que nosotros le proporcionaremos.

Procmail suele estar disponible en cualquier distribución Linux

Las reglas de procmail se refieren al contenido del mensaje en sí (tanto la cabecera como el cuerpo del mismo, **Figura 1**), y nos permitirán eliminar correo propaganda o spam (si no todo, al menos sí gran parte de él), y repartir los mensajes entre los diferentes usuarios de correo, pudiendo tener una sola dirección e-mail para todos los usuarios, pero utilizando algún campo especial para identificar a cada uno.

Instalacion de procmail

Procmail suele estar disponible en cualquier distribución Linux, ya que es un componente muy activo de la misma, al ser utilizado muchas veces por sendmail en la distribución del correo local de la máquina o red. Si no disponemos del mismo es posible obtenerlo de Internet en los ftps de las diferentes distribuciones (*ftp.redhat.com*, *ftp.suse.org*, etc.), e instalarlo vía rpm, dselect/dpkg/apt-get o tar. La versión estable utilizada por el autor es la 3.14-2, aunque la versión variará normalmente en función de la distribución de la que se haya instalado. Tiene una página de ayuda ("*man procmail*", traducida al castellano, como podemos ver en la **Figura 2**, si disponemos de las *man-pages-es*) donde consultar las diferentes opciones y reglas.

Funcionamiento

Existen 2 maneras de instalar procmail. La primera de ellas es que sea utilizado por sendmail a la hora del reparto del correo, y la otra forma se realiza mediante un fichero de configuración de nombre *.forward*, que contendrá la llamada a procmail de forma que éste se ejecute a la llegada de nuevos mensajes al Inbox de dicho usuario. Para saber si tenemos procmail integrado en sendmail se debe buscar la cadena apropiada en */etc/sendmail.cf*.

```
cat /etc/sendmail.cf | grep "^Mlocal"
Mlocal,      P=/usr/bin/procmail, F=(...etc...)
```

Si en dicha línea nos aparece la llamada a procmail, éste ya estará integrado en sendmail y podemos pasar directamente a su configuración propiamente dicha.

El fichero *.forward* (que deberá tener permisos de lectura), sólo será necesario si procmail no es llamado por nuestro MTA (agente de correo), en cuyo caso se necesitará este fichero para llamar a procmail ante cada entrada de mensajes. Dicho fichero se creará en el home del usuario que desea utilizar procmail y contendrá una línea que llame a procmail y que defina el usuario al que debe ir a parar el correo. En caso de que procmail esté integrado ya dentro de sendmail, no necesitaremos crear dicho fichero:

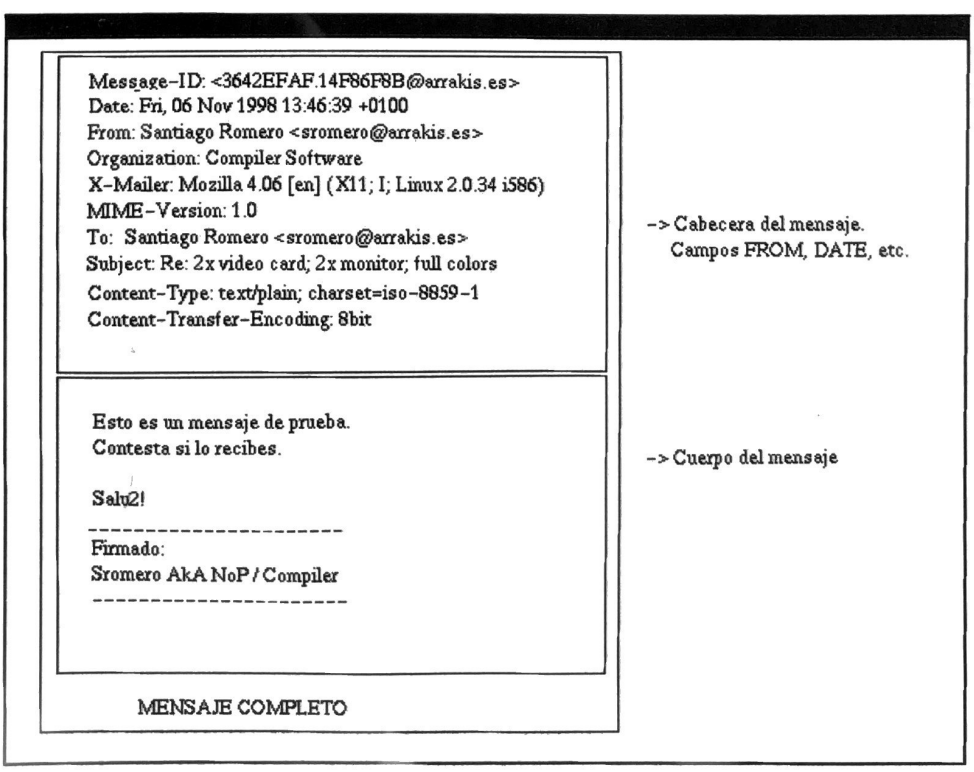

Figura 1. Campos de un mensaje.

```
|IFS=' '&&exec /usr/bin/procmail
-f  || exit 75 #sromero
```

Cuando es llamado (de cualquiera de las 2 formas), procmail lee el mensaje de correo desde la entrada estándar. A continuación abre y lee el fichero *.procmailrc* de nuestro directorio home (si existe), examina las reglas contenidas en dicho fichero y decide el destino del mensaje (que puede ser devuelto, enviado a un usuario en concreto, autorespondido, borrado, etc.).

El conjunto de reglas de *.procmailrc* suelen ser de comprobación y búsqueda de cadenas de texto en el FROM (dirección de origen), SUBJECT (tema del mensaje), BODY (cuerpo del mensaje), así como en el resto de cabeceras del mismo.

Más concretamente, el fichero *.procmailrc* contiene un conjunto de reglas formados por instrucciones condicionales, variables y otras instrucciones. Cuando el mensaje es recibido, éste pasa a ser procesado (procmail = procesador de e-mail), es decir, se leen todas las reglas del fichero y se comprueba si el mensaje cumple alguna de ellas, en cuyo caso se ejecutará el subcódigo pertinente a dicha regla. En caso de no cumplirse ninguna, el mensaje es añadido al final del fichero de mensajes por defecto (especificado como veremos por la variable $DEFAULT).

Visto de una manera sencilla y en lenguaje natural, el fichero *.procmailrc* podría traducirse a algo similar al "lenguaje" siguiente:

```
SI EN EL FROM ENCUENTRAS
usuario@indeseable.com
Borra ese mensaje.

SI EN EL SUBJECT ENCUENTRAS
*CLAVE*PGP*
Envia el fichero
ClavePublicaPGP.txt al
remitente.

PARA EL RESTO DE MENSAJES
Dejarlos en
/home/miusuario/Mail/Inbox
```

Configuracion y ejemplo

Lo primero que vamos a hacer es observar un ejemplo de un *.procmailrc* real antes de comentar las diferentes reglas que pueden usarse, para ir así viendo de qué manera le vamos a indicar a procmail las acciones a realizar con cada tipo de mensaje.

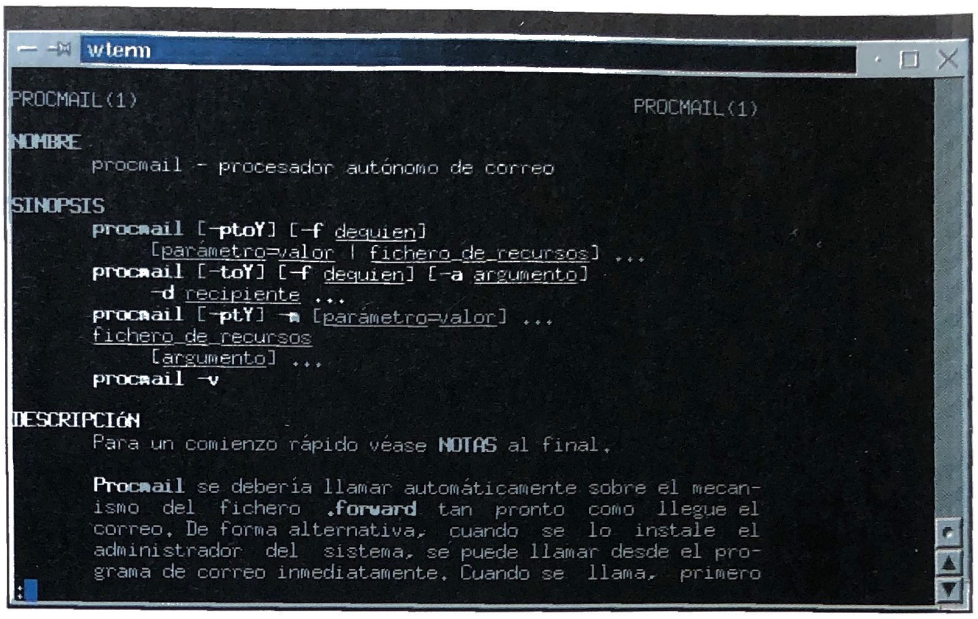

Figura 2. Página man de procmail.

```
# ejemplo de .procmailrc
MAILDIR=$/var/spool/mail
DEFAULT=$MAILDIR/juan
LOGFILE=$MAILDIR/log

:0:
* ^From.*juan
/home/juan/Mail/mensajes_juan

:0:
* ^From.*Josema
!jefe@de.josema

:0:
*
^From.*lista_de_correo@lista.com
/home/juan/Mail/lista_correo_1

:0
* ^Subject:.*prueba
/dev/null
```

En en el ejemplo anterior, las 3 primeras líneas especifican los directorios de correo (MAILDIR), el fichero de entrada de e-mails por defecto (DEFAULT), y un fichero donde procmail indicará paso por paso todo lo que ha hecho en cada sesión (LOGFILE). A continuación se definen 3 reglas (comenzando todas ellas por :0).

La primera de ellas especifica que todos los mensajes que contengan en la cabecera From (el remitente) la cadena *juan*, serán dejados en el fichero *mensajes_juan*. De esta manera, cuando llegue un e-mail cuyo campo From sea *From: juan@ctv.es*, o *From: juan@arrakis.es*, éste será dejado en el fichero mensajes *_juan*. Si esta regla no se cumple (es decir, no aparece la cadena *juan* en el campo From), se pasa a comprobar si se cumple la siguiente regla, que en nuestro caso especifica que todos los mensajes provenientes de josema sean enviados a *jefe@de.josema*, acción realizada mediante el comando ! (cierre de admiración). La penúltima regla nos permite separar el correo de una lista de correo a un fichero diferente de nuestro correo habitual (para así facilitar su lectura en el programa de correo que utilicemos).

Para saber si tenemos procmail integrado en sendmail se debe buscar /etc/sendmail.cf

La última regla especifica que todos los mensajes que contengan la cadena *prueba* en el asunto (Subject), deben ir a para a */dev/null* (also así como la papelera de Linux).

Pero veamos más concretamente la sintaxis de procmail:

```
VARIABLES DE ENTORNO
REGLAS+CONDICIONES
```

Lo primero que veremos serán las diferentes variables de entorno que podemos modificar:

- *DEFAULT*: especifica el fichero donde se almacenará el mensaje que está siendo procesado, si no cumple ninguna de las reglas establecidas en el *.procmailrc*. Este mensaje será añadido al fichero (concatenado), sin borrar los posibles mensajes emplazados ya en dicho fichero. Normalmente este fichero es */var/spool/mail/<usuario>*.
- *MAILDIR*: especifica el directorio donde se almacenarán los mensajes, siendo

generalmente *mail*, *Mail*, */var/spool/mail*, etc., normalmente dentro de nuestro directorio home ($HOME), aunque puede ser cualquier otro directorio en función de cliente de correo que vayamos a utilizar.
- *LOGFILE*: mediante esta variable se le especifica un fichero donde procmail dejará logs de todas las acciones seguidas.
- *INCLUDERC*: permite incluir otro fichero de reglas en cualquier punto del *.procmailrc*.

La asignación de estas variables puede realizarse en cualquier parte del fichero *.procmailrc* (no tiene porqué ser al principio del mismo), de la misma forma que pueden eliminarse si procmail se encuentra una línea con el nombre de una variable y sin asignarle ningún valor (ej: DEFAULT (intro)). En principio, a partir de cada carácter #, el texto hasta fin de línea es considerado como comentario (ignorado por procmail), con la excepción de las líneas en las que se definen condiciones, donde no es posible situar comentarios (sólo antes o tras ellas, pero nunca en la misma línea).

Las reglas de procmail

Todas las líneas que comiencen por :0 o :0:, indican el comienzo de una nueva regla. La diferencia entre :0 y :0: (los 2 puntos finales de la segunda), consiste en que en este segundo caso al escribir en el fichero se bloquea para que no pueda hacerlo otro proceso al mismo tiempo que nosotros y no se corrompa el mismo.
Dentro de cada regla, las líneas que comienzan por un carácter * indican la condición de la regla, mientras que todo lo que siga a las condiciones (pues puede haber más de una condición) se considerarán comandos a ejecutar (excepto las líneas de comentario), tales como en el ejemplo anterior el nombre de fichero (que indicaría a procmail que debe guardar el mensaje ahí), una dirección de correo precedida por ! (que le indica reenvío) o una referencia a */dev/null* (borrado), así como la ejecución de cualquier otro comando Linux que deseemos. Pueden distiguirse 2 tipos de reglas: las que al cumplirse se finaliza el procesado del mensaje (terminales) y las que tras su ejecución se siguen realizando comprobaciones con otras reglas (no terminales).

Una regla tiene un aspecto similar al siguiente esquema:

```
:0 [opciones] [ : ]
* condicion A
* condicion B
* condicion C
etc...
* condición N
comando a ejecutar
```

Tras cada :0 podemos utilizar las siguientes opciones:
- *Opción H* -> La condición se comprobará en la cabecera del mensaje en curso (valor por defecto).
- *Opción B* -> La condición se comprobará en el cuerpo del mensaje.
- *Opción D* -> Hace la comprobación case sensitive, es decir, se distingue entre mayúsculas y minúsculas.
- *Opción i* -> Ignorar cualquier error ocurrido en la regla.
- *Opción c* -> Crea un Carbon Copy, es decir, hace que la regla sea no terminal. Si existe una c en las opciones de la regla, tras finalizar el procesado de la misma se crea una copia del mensaje para que se puedan verificar más reglas.
- *Opciones w y W* -> Espera a que se finalice la ejecución del comando de esa regla para poder recibir el código de salida del mismo. La opción W hace lo mismo que w, pero en caso de ocurrir algún error no emite ningún mensaje sobre el mismo.
- *Opción f* -> Hace a procmail actuar como un filtro. Esto quiere decir que tras ejecutarse las acciones pertinentes sobre el mensaje, se generará un nuevo mensaje de salida que pasará por el resto de reglas del fichero. Esto sirve para modificar campos o valores de un e-mail. Los filtros son, pues, otro tipo de reglas no terminales, pues tras modificar el mensaje seguimos haciéndolo pasar por el resto de reglas.
- *Opción h* -> Hace que en un filtro sólo se filtre la cabecera (que la cabecera se pase al comando especificado).
- *Opción b* -> Hace que en un filtro sólo se filtre el cuerpo o body (que sólo se le pase el cuerpo al comando).
- *Opción E y e* -> Hace que se ejecute la regla sólo si la anterior regla no se ejecutó. En el caso de la e minúscula, sólo se ejecutará una regla si la que la precede en el fichero de reglas se intentó ejecutar pero produjo algún error.
- *Opción A y a* -> Hace que se ejecute la regla sólo si la anterior regla se ejecutó. En el caso de la a minúscula, sólo se ejecutará una regla si la que la precede en el fichero de reglas se ejecutó sin producir errores.
- *Opción r* -> Entrega el mensaje sin realizar correcciones ni comprobaciones.

Como se ha comentado, si no se indica ninguna opción, las condiciones de la regla se aplican a la cabecera del mensaje (H) y pasando como entrada al comando tanto la cabecera (h) como el cuerpo del mensaje (b).

Condiciones

Las condiciones son propiamente la regla en sí, y en caso de que se cumplan se procederá a ejecutar el comando especificado pasándole el mensaje en función de las opciones que hubiésemos seleccionado (por defecto, pasándole la cabecera y el cuerpo). Toda condición empieza por un símbolo * y suelen estar constituidas por expresiones regulares (del tipo de *grep*, *rgrep*, etc.), que consisten en búsquedas de cadenas de carácteres en las diferentes partes del mensaje. En una expresión regular se utilizan los siguientes símbolos:
- *Símbolo '^'* -> Indica el comienzo de una línea.
- *Símbolo '$'* -> Indica el final de una línea.
- *Símbolo '*'* -> Indica cero o más veces.

DEFAULT especifica el fichero donde se almacenará el mensaje que está siendo procesado

- *Símbolo '?'* -> Indica cero o una vez.
- *Símbolo '+'* -> Indica una o más veces.
- *Símbolo '.'* -> Cualquier carácter excepto un *\n* (salto de línea).
- *Símbolo [a-z]* -> Que quede en un rango de caracteres (primero-último).
- *Símbolo [^a-z]* -> Que no quede en un rango de caracteres (primero-último).
- *Símbolo '|'* -> Permite especificar operación O (a | b = 'a' o 'b').

Al comienzo de la línea de condición tambien pueden existir opciones sobre esa condición:
- *Símbolo '<'* -> Permite comprobar si el fichero tiene un tamaño menor al especificado.
- *Símbolo '>'* -> Permite comprobar si el fichero tiene un tamaño mayor al especificado.
- *Símbolo '!'* -> Invierte la condición (se cumplirá cuando la condición sea falsa).
- *Símbolo '$'* -> Realiza susticiones de variables en las expresiones regulares.
- *Símbolo '?'* -> Necesita el resultado que devuelve el programa especificado.

Comandos

Tras cada condición se especifica un comando para que sea ejecutado si la regla se cumple. Distinguimos principalmente cuatro comandos básicos:

1. *Fichero*: hace que procmail añada el mensaje al final del fichero, con otros posibles mensajes.
2. *Directorio*: hace que procmail guarde el mensaje en el directorio con un nombre propio no repetido.
3. *!direccion@email.es*: mediante el carácter '!' podemos enviar el mensaje a la dirección de correo especificada.
4. *|programa*: el carácter '|' permite ejecutar un programa/comando de Linux. La salida del programa saldrá por la salida estándar, aunque se puede redirigir a cualquier lado, con el redireccionamiento estandar de Linux (*>/dev/null*, *>fichero*, etc).

Ejemplos más avanzados

Otro ejemplo de uso de procmail (y muy sencillo) es la autorespuesta de nuestro sistema de correo a determinados e- mails, como envío de la clave PGP (cuando el Subject del mensaje contiene la cadena PGP es usual que nos la estén pidiendo), respuestas automatizadas ante determinados asuntos o Subjects, o autorespuestas para cuando nos vamos de vacaciones:

```
:0:
* ^Subject.*PGP
| (formail -r ; cat
/home/juan/clavepgp.txt) |
sendmail -t
```

En este caso *clavepgp.txt* es un texto con nuestra clave PGP lista para enviar, que pasamos a formail para que componga un mensaje que enviamos con sendmail. El añadir un campo a la cabecera de un mensaje se realiza con *formail* (ver *man formail*), y lo debemos hacer en cada autocontestación:

```
formail -r -A"X-Loop:
nosotros@direccion.es"
```

A la hora de autoresponder también deberemos detectar si ya existe dicho campo y no contestar en ese caso:

```
:0
* !^X-Loop:
nosotros@direccion.es
| (formail -r -A"X-Loop:
nosotros@direccion.es" ;
cat /home/juan/estoy_de_
vacaciones.txt) | sendmail -t
```

Este ejemplo ya es bastante avanzado. También cabe decir que ciertos caracteres especiales en expresiones regulares deberán ser escapados (poner antes de ellos la barra inversa \), para que tengan un significado literal (recordemos que *[aeu]* sería una expresión regular cierta para cualquier cadena que contenga la *a*, la *e* o la *u*, no necesariamente todas ellas. Si quisieramos comprobar si el mensaje contiene la cadena *[aeu]* (con corchetes incluidos) tendríamos que usar la regla *\[aeu\]*, ya que la barra inversa \ le quita el significado especial a los caracteres []. Igualmente debemos proceder para el punto (.), el * y similares. Normalmente para un uso normal de los usuarios, simplemente contendrá algunas reglas sobre separación de mensajes listas de correo a ficheros de texto (ver **Figura 3**), dejando como regla por defecto que los ficheros vayan a parar a *~/Mail/Inbox* o similar, preparados para que los recoja mutt o netscape:

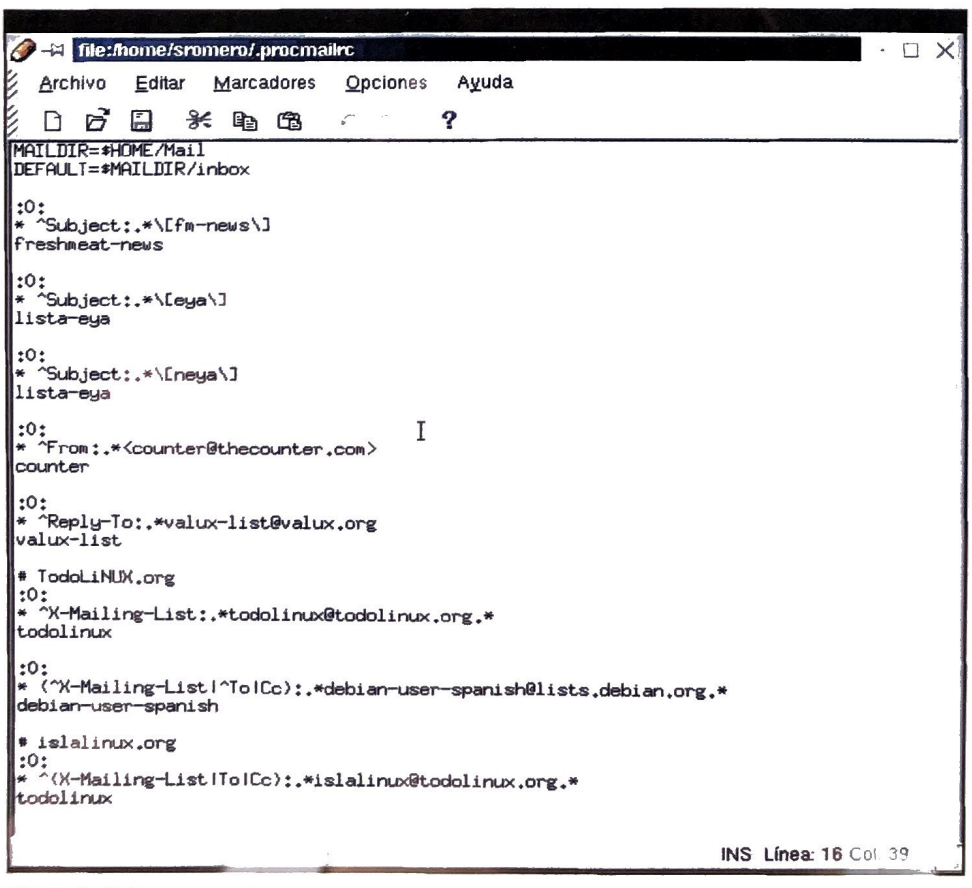

Figura 3. Fichero procmailrc ejemplo.

Todas las líneas que comiencen por :0 o :0: indican el comienzo de una nueva regla

```
# ejemplo de .procmailrc
MAILDIR=$/var/spool/mail
DEFAULT=$MAILDIR/juan
LOGFILE=$MAILDIR/log

:0:
* ^From.*lista-correo-
1@correo.es
/home/juan/Mail/lista-correo1

:0
* ^From:.*fulano@cae.mal
/dev/null
```

Finalmente, es importante recordar que aquellos caracteres que tengan un significado especial para PROCMAIL, deberán ser "escapados" mediante la barra inversa. Por ejemplo, dado que *[y]* tienen un significado especial para el programa (significado de rango o de enumeración de diferentes posibilidades, como en el caso de expresiones regulares), si queremos detectar en el *from* la cadena *]lista-correo[*, deberemos especificar una regla como **^From.*\]lista-correo\[*. En este ejemplo puede apreciarse la manera de introducir caracteres que no queremos que posean significado especial sino literal, mediante la técnica de escapado (anteponiendo \ al carácter).

Finalizando

Mediante el uso de sendmail, fetchmail y procmail, ahora tenemos un total control sobre el correo que entra y sale de nuestra máquina, y estamos preparados para abordar mutt: el cliente de correo más popular de Linux. Como cliente de correo nos permitirá crear e-mails, leerlos, responderlos, reenviarlos, archivarlos, etc., utilizando (en nuestro caso) el utilísimo VIM para la edición de textos. •