La posta con Mutt

Mutt: un segugio come postino

Approfondimenti e personalizzazione

Antonecchia Michele

TELUG

http://www.telug.it
michele@telug.it

Versione: 0.0.8 28 settembre 2001

LaTeX typesetting: Michele Antonecchia
LaTeX editor: ViM version 6.0

Indice

Capitolo 1

Configurazione di base

Ci accingiamo ora a configurare mutt per un utilizzo immediato in modo da poter leggere ed inviare la posta nel minor tempo possibile.
Partiamo dal presupposto che abbiamo installato i pacchetti *fetchmail* e *sendmail*, che di solito sono installati di default dalla maggior parte delle distribuzioni.Per controllare se tali pacchetti sono installati:

```
$ rpm -qa | grep sendmail
$ rpm -qa | grep fetchmail
```

In caso contrario si proceda all'installazione dei pacchetti a partire dal CDRom della stessa distribuzione.

Supponiamo ora che la persona di Michele Antonecchia sia in possesso di due account (POP3) presso *tiscalinet*, ed uno (IMAP) presso un ipotetico *topolug*, con le seguenti caratteristiche:

	1 account	2 account	3 account
user	paperino	pluto	topolino
passw	quiquoqua	osso	basettoni
POP3	pop.tiscalinet.it	pop.tiscalinet.it	
IMAP			mail.topolug.it
SMTP	smtp.tiscalinet.it	smtp.tiscalinet.it	smtp.topolug.it

Tabella 1.1: Account di esempio.

Ricordiamo che il server POP (o in alcuni casi IMAP) è il server dove riceviamo la posta mentre il server SMTP è quello dove spediamo la posta.
N.B: Se si è in possesso di un account presso un altro ISP (ad esempio Libero, Tin, SuperEva, etc ...), basta sostituire i relativi valori dei server POP e SMTP forniti dallo stesso ISP al momento dell'attivazione dell'account.
Inoltre supponiamo che al computer abbiano accesso più persone o che comunque siano presenti più utenti[1] Linux. Noi supporremo di avere gli utenti Linux:

```
disney
michele
```

Fatte le dovute premesse possiamo passare al sodo.

[1]Un utente Linux è identificato dal suo username di accesso

1.1 Fetchmail: Ricevere la posta

Anche se non siamo ancora in grado di leggerla, la posta è conveniente scaricarla, soprattutto se si ha una casella con un alto traffico. In questo modo eviteremo di intasare la nostra casella, che a volte consta di soli pochi mega.

Fetchmail ha anche un tool di configurazione grafico, *fetchmailconf*, che automatizza quello che stiamo per fare. A quanto pare funziona anche egregiamente ma lasciamolo perdere e facciamo le cose a manina, che sicuramente facciamo prima.

Qualsiasi utente è in grado di scaricare la posta per tutti gli altri utenti, ma in questo caso se gli utenti Linux non sono la stessa persona fisica, l'utente che scarica la posta conosce le password degli altri utenti. In tal caso ogni persona deve crearsi un file di configurazione di fetchmail per scaricare la sua posta. Nel caso, invece, che una persona fisica abbia sulla sua postazione Linux accesso a più account (ed è questo il caso che esamineremo) allora potrà scaricare e smistare la posta tra i suoi account direttamente con fetchmail.

Prendiamo ad esempio il caso che si abbiano i tre account di posta di tabella 1.1, e che si voglia smistare il primo ed il terzo all'utente Linux `disney` mentre il secondo all'utente Linux `michele` .

Dato che preferisco connettermi ad internet tramite l'utente `michele` creo il file di configurazione `.fetchmailrc` (nota il punto) nella directory home di `michele` ,`/home/michele/.fetchmailrc` .

Quindi facciamo prima un bel `cd` per posizionarci nella nostra directory home, creiamo il file `.fetchmailrc` e ci scriviamo:

```
poll pop.tiscalinet.it with proto POP3 timeout 60
user paperino there with password quiquoqua is disney here

poll pop.tiscalinet.it with proto POP3 timeout 60
user pluto there with password osso is michele here

poll mail.topolug.it with proto IMAP timeout 60
user topolino there with password basettoni is disney here options keep
```

Salviamo, usciamo e cambiamo i permessi al nostro file altrimenti fetchmail non funziona.

```
$ chmod 600 .fetchmailrc
```

Notiamo che l'ultimo account differisce dagli altri per un particolare. Innanzitutto utilizza il protocollo IMAP anzichè POP, poi compare una opzione, "keep". Tale opzione serve a non fare cancellare i messaggi sul server. Sarebbe buona idea se mentre fate queste prove metteste tale opzione a tutte le vostre caselle di posta, in tal modo nulla andrebbe perso, poi quando si è sicuri di aver raggiunto la stabilità della configurazione si può sempre togliere la stringa "`options keep`".

Per scaricare la posta a questo punto basta eseguire il comando:

```
$ fetchmail -v
```

oppure semplicemente

```
$ fetchmail
```

Se invece si vuole che fetchmail provi a scaricare la posta ogni 10 minuti (600 secondi), si lancia il comando:

```
$ fetchmail -d 600
```

N.B.: Le nuove versioni di fetchmail hanno cambiato la gestione del POP/IMAP. Se usate una versione recente (superiore a 5.7) allora dovete lanciare fetchmail come di seguito

```
$ fetchmail --auth password
```

Ovviamente le opzioni sono cumulative e potreste aver bisogno di dare il comando più complesso:

```
$ fetchmail -d 600 -v --auth password
```

La posta scaricata va a finire (secondo l'esempio) in:

```
/var/spool/mail/michele
/var/spool/mail/disney
```

N.B.: fetchmail può essere utilizzato con qualsiasi client di posta elettronica, sia testuale che grafico, basta avere l'accortezza di non configurare il client per accedere al server POP, ma dicendogli di accedere direttamente alla mailbox locale.

1.2 Mutt: Leggere la posta

Finalmente ... non direi ;-) ... possiamo leggere la posta.
Mutt é un programma leggero ma molto efficiente per leggere la posta da console. La sua flessibilità lo pone come uno strumento molto utile per un uso professionale per leggere la posta. Inoltre forniscedelle avanzate feature quali il "key bindings", il "keybord macros", il threading dei messaggi, la ricerca tramite espressioni regolari ed altro.
In questa sezione, però, spiegheremo soltanto come realizzare una configurazione di base per mutt, per gli approfondimenti basta riferirsi ai capitoli successivi dove verrà presa in analisi una configurazione un pò più personalizzata.

1.2.1 Il file .muttrc

Creiamo le directory `Mail, Mail/Friends, Mail/Lists`:

```
$ mkdir /home/tuo-utente/Mail
$ mkdir /home/tuo-utente/Mail/Friends
$ mkdir /home/tuo-utente/Mail/Lists
```

ed il file `Mail/.mail_aliases` [2]:

```
$ cd /home/tuo-utente/Mail
$ touch .mail_aliases
```

[2] Occhio al punto

copiamo il file `.muttrc`, che troverete allegato a questo manuale [3]

```
$ cp dove-si-trova-il-file/.muttrc /home/tuo-utente
```

lo editiamo ed andiamo a cambiare i valori che si trovano nella sezione `Dati personali`, ed in particolare modifichiamo le seguenti voci:

```
set hostname="inventato.it"
set realname = "Michele Antonecchia"
set spoolfile='/var/spool/mail/michele'
```

N.B.: `/var/spool/mail/michele` *é il file di spool dove viene scaricata la posta in arrivo, ad esempio, da fetchmail. Se non si utilizza fetchmail impostare la chiave "*`set spoolfile` *in modo da puntare allo spool in cui scaricate la posta con il vostro programma.*

A questo punto prima di eseguire mutt bisogna scaricare la posta.

```
$ fetchmail -v
$ mutt
```

Se tutto é andato per il verso giusto, e se avete ricevuto posta, compariranno i vostri tanto desiderati messaggetti.

1.2.2 Breve panoramica sulla gestione dei messaggi

Una volta in possesso di almeno un messaggio possiamo passare ad un mini tutorial sulla gestione dei messaggi.

In qualsiasi momento è possibile entrare nel menu di aiuto digitando semplicemente il carattere "?".

Spostarsi nei menu

Innanzitutto vediamo come spostarci nei menu una volta lanciato mutt.

j o Down	next-entry	si sposta al successivo messaggio
k o Up	previous-entry	si sposta al precedente messaggio
z o PageDn	page-down	va alla prossima pagina
Z o PageUp	page-up	va alla pagina precedente
= o Home	first-entry	salta al primo messaggio
_* o End	last-entry	salta all'ultimo messaggio
q	quit	esce dal menu corrente
?	help	attiva la lista di tutti i comandi

Tabella 1.2: Comandi di movimento

I comandi sono simili a quelli utilizzati per ELM.

[3] Il file `.muttrc` non è disponibile nella versione online ma solo nelle versioni scaricabili. I link per i download sono presenti in testa alla versione online.

Creare un messaggio

Per creare un nuovo messaggio basta premere il tasto "m". Se invece si vuole rispondere al messaggio il tasto è "r".

Compariranno in basso, in ordine, i campi To: e Subject: dove inserire il destinatario ed il soggetto come si fa con tutti gli altri client di posta.

Una volta inviato anche il Subject:, mutt aprirà l'editor, [4] che di default è vim, e si potrà editare il messaggio.

Finita la creazione del messaggio, si salva e si esce. Mutt chiederà cosa si vuole fare del file appena salvato. Premendo "y" il messaggio viene spedito allo spool di uscita[5].

N.B: Se si vuole allegare un file premere "a" prima di "y", scegliere il file ed infine premere "y".

Grazie all'opzione `set autoedit` non impostata, personalmente, ritengo sia più comodo scrivere i campi To: e Subject: ed utilizzare gli aliases [6]. Nel caso si decommenta tale opzione nel file `.muttrc`, tali campi dovranno essere aggiornati in sede di editazione del messaggio, spostandosi con i tasti freccia sui relativi spazi lasciati vuoti.

Creare gli aliases

Gli aliases sono una sorta di rubrica degli indirizzi e tale rubrica è il file `.mail_aliases` . Pertanto gli aliases sono molto utili quando non si vuole scrivere per intero l'indirizzo di posta elettronica.

Per creare un aliases ci sono due strade:

- Si edita direttamente il file `.mail_aliases` e si scrive una cosa del tipo:

  ```
  alias nome_corto Vero Nome <email@isp.it>
  ```

- Oppure si seleziona un messaggio del mittente del quale si vuole creare un alias, e si preme il tasto "a". A questo punto parte la procedura automatica per creare un alias del tutto identico al precedente.

Quindi, quando si crea un messaggio, al momento di scrivere il campo To: o si scrive il "nome_corto" oppure si preme il tasto "TAB" per vedere l'intera lista degli aliases.

Per mandare il messaggio a più persone o si scrive nel campo To: una sequenza di nome_corto1,nome_cort2,..., oppure si preme il tasto "TAB" si selezionano, con "enter", tutti i destinatari e si esce con "q".

Dopo di che si procede nella normale procedura di creazione.

Uscire da mutt: Archiviare i messaggi

Quando si esce da mutt viene chiesto se archiviare i messaggi letti oppure lasciarli ancora in visione. Se si risponde affermativamente i messaggi verranno spostati dallo spool di entrata e salvati in `~/Mail/mbox` .

[4] Nel prossimo paragrafo è spiegato come cambiare editor

[5] Sarà necessario eseguire sendmail per spedire definitivamente il messaggio. Se invece si è in linea la spedizione è automatica.

[6] Si rimanda alla sezione più avanti

Per rileggere i messaggi salvati bisogna caricare una mailbox, che nel caso in esame è una sola, con il tasto "c" seguito da un successivo "?". Il tasto "c" serve a dire che si vuole caricare una mailbox, il tasto "?" per visualizzare le mailbox a disposizione.

Con il file di configurazione da noi creato tutte le email verranno salvate in `~/Mail/mbox` .

Cambiare editor

Di default nel file `.muttrc` ho settato `vim` come editor. Se volete invece utilizzare un altro editor dovete cercare la chiave `set editor="vim +11"` nel file `.muttrc` alla sezione "Editor e Stampa" e cambiarla. Ad esempio per utilizzare `emacs` la chiave deve essere

```
set editor="emacs -nw"
```

oppure se volete utilizzare `nedit`

```
set editor="nedit +11"
```

N.B.: Se viene settato un editor grafico non sarà possibile scrivere un messaggio da console ma soltanto in ambiente X.

Inoltre ricordo che "+11" significa che l'editor deve posizionare il cursore alla undicesima riga, che per come è fatto il nostro `.muttrc` significa appena sotto gli header ;-)

La firma

È abitudine diffusa contrassegnare i propri messaggi con una "firma"[7]. Essa puo' contenere le cose più assurde (frasi, poesie, ascii art, indirizzi, etc ...) ma deve essere "formattata".

Per formattata si intende che la firma vera e propria deve essere delimitata superiormente dai caratteri `--` cioè `trattino trattino spazio`. Lo spazio è importante.

Quando un client di posta (mutt) o un newsreader (slrn) incontrano tale delimitatore sanno che ciò che segue è una firma e la contraddistingono con un diverso colore.

La firma è deposta nel file `/home/michele/.signature.mutt` in accordo con la chiave `set signature` . Infatti in `.muttrc` essa è settata come

```
set signature="~/.signature.mutt"
```

e pertanto il suo contenuto va cambiato secondo le esigenze.

Ad esempio il mio file `.signature.mutt` è così fatto:

```
        Provare che ho ragione significherebbe riconoscere
        che posso aver torto.
mail to: michele@telug.it
website: http://www.telug.it
```

In realtà io sono pigro, e preferisco mettere i saluti in automatico. Allora per fare questo utilizzo una signature un pochetto diversa[8]

[7] Di solito é buona norma che la firma non superi le 4 righe

[8] Anche questa firma rispetta le 4 righe concesse dalla Netiquette.

```
Ciao, Michele.
--
        Provare che ho ragione significherebbe riconoscere
        che posso aver torto.
mail to: michele@telug.it
website: http://www.telug.it
```

Ma attenzione. In `.muttrc` esiste la chiave `set sig_dashes` che serve a mettere il delimitatore `--` in automatico. Per utilizzare in modo corretto la seconda firma, devo disattivare questa opzione (altrimenti "Ciao, Michele" diventa parte della firma) nel seguente modo

```
set sig_dashes=no
```

ed inserire il delimitatore[9] direttamente nel file della signature.

Per controllare che la firma sia ben formattata assicuratevi che quando scrivete un messaggio essa sia colorata.

1.3 Sendmail: Spedire la posta

Configurare sendmail per un uso in una intranet è forse una cosa complicatissima[10] ma per noi che ce ne stiamo belli seduti a casa davanti al nostro pinguino solitario è, invece, la cosa più semplice di questo mondo.

Ci logghiamo come root ed editiamo il file `/etc/sendmail.cf`, senza farsi impressionare dalla mole di informazioni che contiene. Cerchiamo le righe

```
# "Smart" relay host (may be null)
DS
```

e le trasformiamo in

```
# "Smart" relay host (may be null)
DSsmtp.tiscalinet.it
```

Mi raccomando, occhio agli spazi, che non ci sono. Adesso basta soltanto riavviare sendmail con un `restart` che, a seconda delle distribuzioni, si traduce nell'eseguire come utente `root` uno dei comandi

```
# /etc/rc.d/init.d/sendmail restart
# /etc/init.d/sendmail restart
```

Per spedire la posta basta eseguire:

```
$ sendmail -q -v
```

se si vuole vedere in modo "verbose" ciò che sendmail sta facendo, oppure semplicemente

```
$ sendmail -q
```

[9] Attenti sempre allo spazio

[10] Se si ha bisogno di una configurazione più complicata di sendmail i lavori sono in corso

N.B.: Non sempre un utente normale, cioè non root, può spedire la posta.
Soluzione: O ci logghiamo da root e spediamo la posta come gia detto oppure creiamo un link simbolico a sendmail in una dir del nostro path. In quest'ultimo caso ci logghiamo come root e creiamo il link:

```
# ln -s /usr/sbin/sendmail /home/disney/bin/
```

In questo modo andiamo un pochetto in barba con la sicurezza ma non crediate che sia meno sicuro che aprire una shell di root (e poi non siamo paranoici).
Ma c'è un'altra soluzione ancorpiù interessante: usare `sudo`. Ma per questo vi rimando alla sezione dei `Tips & Tricks` a pagina 37

1.3.1 Un accorgimento in più

Liberamente tratto dal `Sendmail-Address-Rewrite-howto`.

Chi ha fretta di mettere in funzione mutt e leggere subito la posta può saltare, per il momento, questa sezione. Raccomando vivamente, peró, di tornarci su in seguito in quanto di elevato interesse.

Con la configurazione appena fatta di sendmail nasce un problema. Quando inviamo la posta il nostro MTA riscrive negli header del messaggio un campo from con il nome della nostra macchina. Chiaramente la macchina ha un nome fittizio e non e' raggiungibile da internet, cosi quando un nostro corrispondente scarica l'e-mail il suo sendmail o fetchmail vedono che c'e' un campo from con dominio inesistente, credono che si tratti di spam e non scaricano il messaggio che noi gli abbiamo inviato.[11]
A questo punto è necessario installare i pacchetti `m4` e sendmail-cf [12] dal cd della nostra distribuzione oppure scaricarli da internet.

Apriamo il file `sendmail.mc` che, se non sappiamo dove si trova, lo rintracciamo con il comando

```
# find / -name sendmail.mc
```

e vediamo se contiene le seguenti righe

```
include(/usr/lib/sendmail.cf/m4/cf.m4)
VERSIONID(‘sendmail.mc - roessler@guug.de’)
OSTYPE(debian)
define(‘ALIAS_FILE’,‘/etc/mail/aliases’)
```

ma soprattutto se i file che vengono richiamati esistono e sono posizionati nel percorso specificato.

Come primo file viene incluso il file `cf.m4`. Questo m4 macro file contiene molte macro definitions per il resto del file. La macro OSTYPE è usata per assegnare alcuni parametri di default ad hoc per la nostra distribuzione. In particolare se usate una Debian la macro OSTYPE richiama il file `ostype/debian.m4` il quale contiene specifiche stringhe di configurazione per la vostra distribuzione. Se invece non usate una distribuzione Debian, molto probabilmente il file che a voi interessa e `ostype/linux.m4` e quindi dovrete sostituire la

[11] Sono comunque casi particolari
[12] Nella Slackware questo pacchetto si chiama `smailcf`

parola "debian" con "linux"[13]. `ALIAS_FILE` comunica a sendmail dove guardare per la lista degli aliases.

Invece le seguenti righe dicono a sendmail di usare la feature `genericstable` e dove cercare i file necessari per utilizzarla.

```
FEATURE(masquerade_envelope) FEATURE(genericstable, ‘hash -o \\
       /etc/mail/genericstable’)
GENERICS_DOMAIN_FILE(‘/etc/mail/genericsdomain’)
```

La feature `masquerade_envelope` dice a sendmail di applicare la riscrittura degli header per "coprire" il mittente del messaggio. Questo è l'indirizzo di posta con il quale un mail delivery esterno dirotterà la spedizione dei report failure e dei messaggi di warning.

Adesso abbiamo bisogno di definire lo `smart host`, cioè, una macchina con la quale trattare la posta in uscita dal nostro sistema. Da notare che il nome di questa macchina può essere diversa da quello dell' ISP o del server POP/IMAP[14]. Il codice da inserire nel file di configurazione è:

```
define(‘SMART_HOST’,‘smtp.tiscalinet.it’)
```

In realtà questa riga non farà altro che fare la modifica che nella sezione precedente è stata fatta di forza bruta a mano.

Le ultime due righe includono le definizioni del "mailer", che sono necessarie a sendmail per trovare il modo di trattare vari tipi di mail.

```
MAILER(local)
MAILER(smtp)
```

Per generare il file di configurazione vero e proprio `sendmail.cf` a partire da `sendmail.mc` digitare i seguenti comandi come root:

```
# m4 sendmail.mc > _sendmail.cf
# mv -f _sendmail.cf sendmail.cf
```

Da notare la tecnica di scrivere l'output del processore m4 in un file temporaneo e poi spostarlo nel proprio posto. Ciò aiuta a prevenire il fatto che sendmail possa leggere parzialmente il file di configurazione che man mano viene generato.

Un grazie a Paolo Pierasanti (anche lui componente del TeLUG) il quale mi ha fatto notare l'inghippo e aiutato a risolverlo.

1.4 Postfix

La instabilitá e la pesantezza che hanno sempre accompagnato sendmail hanno fatto si' che molte distribuzioni hanno deciso di utilizzare di default altri MTA. Tra i più stabili si distinguono QMail e Postfix. Il primo ha una struttura modulare ma molto complessa, il secondo più semplice e più facile da configurare.

Il tempo e' tiranno e non ne ho ancora a sufficenza per completare questa sezione. Comunque sia il seguente `/etc/postfix/main.cf` può bastare per fare andare la baracca senza troppi intoppi.

[13] Di solito questo viene fatto dall'installazione del pacchetto stesso e quindi potrà non essere necessaria questa operazione

[14] Infatti stiamo parlando del server SMTP

```
queue_directory = /var/spool/postfix
command_directory = /usr/sbin
daemon_directory = /usr/lib/postfix
mail_owner = postfix
mail_owner = postfix
default_privs = nobody
myhostname = bonovox.it
mydomain = thefly.it
alias_maps = hash:/etc/postfix/aliases
mail_spool_directory = /var/spool/mail
mailbox_command = /usr/bin/procmail -a $DOMAIN -d $LOGNAME
relayhost=[smtp.tiscalinet.it]
smtpd_banner = $myhostname ESMTP $mail_name ($mail_version) /
      (Linux-Mandrake)
local_destination_concurrency_limit = 2
default_destination_concurrency_limit = 10
debug_peer_level = 2
debugger_command =
     PATH=/usr/bin:/usr/X11R6/bin
     xxgdb $daemon_directory/$process_name $process_id & sleep 5
```

Prossimamente la configurazione di postfix completa :-) e tante altre cose.
A presto.

1.5 Procmail: mutt mette le ali

Di certo una configurazione come quella data nel capitolo precedente è alquanto spartana e poco efficiente. Basti pensare che l'iscrizione ad una sola mail-list potrebbe sconvolgere l'equilibrio tra gestione ed efficienza. A volte si vole persino raccogliere da parte la posta che arriva dall'amante in modo da tenerla separata da quella ricevuta dalla fidanzata :-).
Un singolo file di mailbox, quale mbox, non può ,per sua natura, rispondere alle nostre esigenze ed è per questo che dobbiamo chiedere aiuto ad altre fonti.
Anche se lo stesso mutt consente di smistare la posta in più mailbox, in questa sezione analizzeremo l'uso di procmail per svolgere questa mansione. In questo modo ripensamenti futuri su mutt non impediranno un passaggio veloce ad un altro lettore di posta.

Procmail è un progamma molto utile per gestire la posta scaricata da fetchmail o che arriva in tempo reale alla nostra postazione (connessioni 24/24h). La sua funzione è quella di leggere lo spool di entrata, quali

```
\var\spool\mail\disney
\var\spool\mail\michele
```

identificare, tramite delle regole che di seguito vedremo, i messaggi, e creare diverse mailboxes di tipo standard[15].

La prima cosa da fare è, quindi, quella di obbligare i messaggi a passare sotto la tutela di procmail, dire a procmail dove deve leggere lo spool dei file, dare le regole e smistare i messaggi in base ad esse. Un esempio chiarirà il discorso.

[15]Per tipo standard si intende che ogni messaggio è accodato al precedente e separato da due linee vuote

1.5.1 Il file .forward

Per prima cosa bisogna creare il file `.forward`[16] nella propria home directory.

```
$ cd ~
$ vim .forward
```

e scriverci la seguente riga **senza andare a capo**[17]

```
"| IFS=' ' && p=/usr/bin/procmail && test -f $p && exec $p -Yf- ||
exit 75 #michele"
```

Tale file serve ad indicare al deliver (il programma usato dal nostro MTA per consegnare la posta in locale) se il messaggio di quell'utente[18] deve essere reindirizzato, creandolo con il contenuto su indicato. In definitiva si ha l'effetto di passare il messaggio in arrivo al comando specificato nella stringa.

In mancanza di tale file, i messaggi verranno smistati senza alcun tipo di operazione svolta su di essi.

1.5.2 Configurazione di procmail: le regole

Come abbiamo visto per sendmail, mutt e fetchmail, anche procmail ha un file di configurazione: `.procmailrc` [19]. Adesso analizzeremo un file `.procmailrc` di esempio per capirne meglio la struttura. [20]

```
# Please check if all the paths in PATH are reachable, remove the ones that
# are not.

PATH=$HOME/bin:/usr/bin:/usr/ucb:/bin:/usr/local/bin:.
MAILDIR=$HOME/Mail      # E‘ la dir dove conserviamo i messaggi
DEFAULT=/usr/spool/mail/michele
LOGFILE=$MAILDIR/.from
LOCKFILE=$HOME/.lockmail

:0                          # Tutto cio' che viena da thf
* ^From:.*thf@somewhere.someplace
$MAILDIR/todd               # andra' in $MAILDIR/todd come unico file

:0                          # Tutto cio' che viena dal dominio uunet
* ^From:.*@uunet
$MAILDIR/uunetbox           # andra' in  $MAILDIR/uunetbox come unico file

:0                          # Le lettere di Henry
* ^From:.*henry
```

[16] Occhio sempre al punto
[17] Mi raccomando. Senza andare a capo.
[18] In questo caso l'utente Linux `michele`.
[19] Il punto c'è sempre :-)
[20] Nella dir `/usr/doc/procmail-3.14/examples/` ci sono alcuni file di esempio

```
henries                         # andranno in $MAILDIR/henries/nome-file

:0                              # "Tutte" le mail che contengono SPAM
* ^Subject.*MAKE.*MONEY.*FAST.*
\dev\null                       # Vanno nel pozzo nero

:0                              # La mail-list di zio-paperone
* ^From:.*mailing-list@*zio.paperone.*
$MAILDIR/zio-paperone           # sara‘ conservata nel folder zio-paperone


# Anything that has not been delivered by now will go to $DEFAULT
# using LOCKFILE=$DEFAULT$LOCKEXT
```

Le prime due righe hanno il compito di definire le variabili `PATH` e `MAILDIR` che servono a procmail. Importante è `MAILDIR`. Essa definisce il percorso della directory che conterrà le mailbox dove verranno smistati i messaggi. Tali mailbox hanno lo stesso formato del file di spool.

Tralasciamo per ora le altre righe e soffermiamoci sui "contenitori".

Un contenitore è una parte del file .procmailrc. Esso inizia con il set di simboli "`:0`" (duepunti zero) e finisce dove inizia un nuovo recipiente o alla fine dello stesso file. Tale recipiente segue delle regole:

- A `:0` possono seguire dei particolari flag
  Alcuni flag sono:
  - c - Processa il messaggio senza spostarlo dallo spool di entrata
  - B - La regola si applica al corpo del messaggio
  - H - (default) La regola si applica all'intestazione del messaggio
  - D - La regola è case sensitive
- Dopo `:0` segue la regola
  La regola inizia sempre per * e contiene un espressione regolare da confrontare con il contenuto del messaggio (l'intestazione e/o il corpo, a seconda del flag attivato)
  - Il carattere ˆ "apice" significaall'inizio della riga
  - Il carattere . significa qualsiasi carattere
  - Il carattere * significa che l'ultimo carattere specificato può essere contenuto anche più volte
- Segue il percorso del folder

Quindi la regola

```
:0
* ^From:.*mailing-list@*zio.paperone.*
```

va letta come: All'inizio della riga cerca la parola From: negli header del messaggio, seguita da qualsiasi numero di caratteri (anche nessuno) e dall'espressione mailing-list@zio.paperone a sua volta seguita da una qualsiasi quantità di caratteri, fino alla fine riga.

Pertanto entrambe i messaggi:

```
From: La lista della spesa <mailing-list@zio.paperone>
From: Le consegne della settimana <mailing-list@zio.paperone>
```

saranno processati da quella particolare regola

Analizziamo ora una particolare regola:

```
:0 c
* ^Subject.*URGENTE*
! michele@postaprivata.it
```

In tal caso tutti messaggi dichiarati urgenti saranno reindirizzati (forwardati è orribile) a quell'indirizzo di posta, dove si presume venga scaricata quotidianamente ;-)

Pertanto il flag "c" serve a non perdere il messaggio, mentre "!" serve a fare il forward del messaggio. Se non si utilizzava il flag "c" il messaggio veniva reindirizzato e tolto dallo spool definitivamente.

Adesso vediamo un'altra particolarità:

```
:0
* ^From:.*thf@somewhere.someplace
$MAILDIR/todd

:0
* ^From:.*thf@somewhere.someplace
todd
```

Le due regole NON sono uguali. La prima salva i messaggi in un unica mailbox, accodandoli. La seconda li salva nella directory todd assegnandogli un nome unico. Se al nome della directory viene aggiunto ./ i messaggi verranno salvati con un nome di file numerico, progressivo.

1.5.3 Un archiviatore automatico

La potenza di procmail non si sofferma soltanto a smistare la posta nei modi più astrusi, procmail è capace anche di passare un messaggio ad un programma. Se in .procmailrc inseriamo anche[21] le variabili

```
DATE=`date +%y-%m`
ARCHIVEDIR=$MAILDIR/archive
```

Una regola come questa

```
:0 c
* ^From:.*mailing-list@zio.paperone
| gzip >> $ARCHIVEDIR/zio.paperone.$DATE.gz
```

[21] Si intende che le variabili qui introdotte si aggiungono a quelle precedenti

Ha l'effetto di archiviare automaticamente, mese per mese[22], tutta la posta proveniente dall'indirizzo mailing-list@zio.paperone.

Infatti la variabile `DATE` serve per determinare il formato della data come anno-mese, mentre la variabile `ARCHIVER` specifica il percorso dove salvare l'archivio. Il flag c impedisce che il messaggio sia archiviato senza poterlo leggere.

N.B.: Tale regola deve precedere qualsiasi altra regola che riguarda tale indirizzo e che non abbia attivo il flag "c". Altrimenti la regola che precede quella di archiviazione cancella il messaggio dallo spool e l'archiviazione non avviene.

1.5.4 Recipienti innestati e azioni complesse

A volte si vuole operare su un singolo messaggio se esso contiene più informazioni. Cioè se esso risponde a più criteri contemporaneamente.

Per questo è possibile creare recipienti innestati.

```
:0
* ^From:.*mailing-list@*zio.paperone.*
{
:0 c
* ^Subject.*URGENTE*
! michele@postaprivata.it
$MAILDIR/zio-paperone
}
```

In questo caso quando il campo From AND Subject rispondono alle rispettive regole il messaggio viene reindirizzato e salvato, grazie al flag "c" nel suo folder.

Cose più complicate si possono fare adottando qualche trucco, ad esempio racchiudendo i comandi in parentesi tonde.

Inoltre le regole devo stare su una singola riga o mandate a capo con un *backslash*. Ad esempio la regola

```
DATE=`date +%y-%m`
ARCHIVER=$MAILDIR/archive
SPAMDIR=$MAILDIR/spam

* ^Subject.*MONEY*
| (echo "Spam ricevuto in data `date`; \
cat) | gzip >> $SPAMDIR/spam.$DATE.gz
```

equivale a

```
DATE=`date +%y-%m`
ARCHIVER=$MAILDIR/archive
SPAMDIR=$MAILDIR/spam

* ^Subject.*MONEY*
| (echo "Spam ricevuto in data `date`; cat) | gzip >> $SPAMDIR/spam.$DATE.gz
```

[22] se in `DATE` si inserisce anche il giorno, la posta sarà archiviata giorno per giorno

ed in particolare quando il subject contiene la parola MONEY, procmail fa precedere il messaggio dalla frase *Spam ricevuto in data 00-09* e lo archivia tramite gzip nel folder spam.`$DATE.gz` .

1.5.5 Rivediamo il file `.muttrc`

Utilizzando procmail [23] è necessario fare alcune modifice al nostro `.muttrc`.

In particolare bisogna dire a mutt quali sono le mailboxes che hanno ricevuto le nuove mail e, nel caso, visualizzare quali hanno effettivamante hanno un nuovo contenuto.

Le prossime operazioni si riferiscono alle mailboxes create tramite `procmail`. Ad esempio se tramite `procmail` abbiamo dato la seguente regola

```
:0
* ^Subject.*ml-linux.*
$MAILDIR/Lists/ml-linux

:0
* ^From:.*@topolug.it.*
$MAILDIR/Lists/topolug
```

allora nel nostro `.muttrc` dobbiamo inserire le seguenti righe nella sezione `Mailboxes to watch for new mail`

```
mailboxes ‘echo $HOME/Mail/Lists/*‘
mailboxes $HOME/Mail/mbox
```

In questo modo sia la mailbox `topolug` che `ml-linux` saranno inserite, tramite l'aterisco, come mailbox "capaci di contenere posta".

Ovviamente nella sezione delle mailboxes è necessario "linkare" tutte le mailboxes create con `procmail`. In generale una sezione come la seguente è in grado di provvedere a tutto.

```
mailboxes ‘echo $HOME/Mail/Lists/*‘
mailboxes ‘echo $HOME/Mail/Friends/*‘
mailboxes $HOME/Mail/mbox
```

Comunque sia avete gli strumenti necessari per personalizzare a dovere le vostre mailboxes. Ora è però necessario dire a mutt di leggere le mailboxes, e questo si fa aggiungendo il flag y quando lanciamo mutt

```
mutt -y
```

In questo modo al lancio di mutt verranno visualizzate le mailboxes specificate e quelle che contengono nuova posta saranno contrassegnate dalla N.

Infine, quando abbiamo finito di leggere una mailbox, premendo il tasto "c" ci verrà proposto di passare alla successiva mailbox che contiene un messaggio nuovo, se esiste, oppure, in caso che abbiamo già letto tutti i messaggi, ci verrà proposto di scegliere la mailbox invitandoci a premere il tasto "?".

[23] O in generale quando si usano più mailbox

Capitolo 2

Gestione avanzata

Fino ad ora abbiamo fatto una panoramica generale e ad alta quota sull'argomento per il semplice fatto che ci premeva mettere in piedi una struttura veloce e funzionale. Adesso pero' possiamo divertirci a scoprire qualcosa in più e capire perchè mutt é un attrezzo funzionale, versatile ed efficiente.

Da sottolineare che la mappatura della tastiera è quella di default. Nel senso che l'associazione tasto-funzione è quella di default. Pertanto se lavorate con mutt in una postazione configurata da altri, il tasto p (ad esempio) potrebbe non fare quello che questo manuale dice.

2.1 I comandi e i flag

Questa è la sezione meno piacevole alla lettura ma che comunque serve a poter svolgere le più comuni e frequenti operazioni.

Abbiamo giá accennato a pagina 5 ai basilari comandi di movimento e pertanto mi limiterò, qui, soltanto a introdurre i restanti.

Come in tutti gli altri clienti di posta, mutt ha due modalità di lettura: l'Index ed il Pager. Si è nel primo quando si vede la lista dei messaggi contenuti in una mailbox. Si è nel pager, invece, quando si visualizza il contenuto del messaggio.

2.1.1 I comandi dell'Index

`c`	`cambia mailbox`
`ESC c`	`cambia cartella in modalita‘ sola lettura`
`C`	`copia il messaggio corrente in un'altra mailbox`
`ESC C`	`decodifica un messaggio e copialo in un'altra mailbox`
`ESC s`	`decodifica un messaggio e salvalo in un'altra cartella`
`D`	`cancella i messaggi che corrispondono ad un pattern`
`d`	`cancella il messaggio corrente`
`F`	`marca il messaggio come importane`
`l`	`visualizza solo i messaggi che corrispondono ad un pattern`
`N`	`marca il messaggio come nuovo`
`o`	`cambia il corrente metodo di ordinamento`
`O`	`inverti il metodo di ordinamento`
`q`	`salva ed esci`

```
s               salva un messaggio
t               commuta il tag di un messaggio
ESC t           commuta il tag sui messaggi di un intero thread
u               ripristina un messaggio (undelete)
v               vedi l'allegato
x               annulla i cambiamenti ed esci
<Return>        visualizza un messaggio
<Tab>           salta al prossimo nuovo messaggio
@               visualizza l'indirizzo completo dell'autore
$               salva i cambiamenti alla mailbox
/               cerca
ESC /           ricerca inversa
^L              pulisci e "riscrivi" lo schermo
^T              commuta il tag dei messaggi che corrispondono
 ad un pattern
^U              ripristina i messaggi che corrispondono ad
 un pattern (undelete)
```

2.1.2 I comandi del Pager

```
<Return>        scorri in gi\`u una linea
<Space>         visualizza la prossima pagina
           (o il prossimo messaggio se alla fine del messaggio corrente)
-               torna alla pagina precedente
n               visualizza il prossimo messaggio
?               mostra le associazioni tasto funzione (keybindings)
/               cerca secondo un'espressione regolare (pattern)
\               commuta la colorazione della ricerca
```

2.2 Mailing List e Friends List

Procmail non è strettamente necessario per la gestione di più mailboxes. Mutt riesce anche lui in modo egregio a soddisfare tale esigenza. A volte si preferisce Procmail in quanto se si desidera cambiare il programma per leggere la posta non si perdono tutte le regole, che a volte sono davvero tante. Resta comunque da dire che le mailboxes che usa mutt sono standard ed esportabili su quasi tutti i lettori di posta (seri :-).

Comunque l'argomento è stato già trattato nella sezione 2.5, alla quale si rimanda. Inoltre nella sezione 2.5.5 troverete le modifiche da apportare a `.muttrc` in modo da poter gestire più comodamente le mailbox delle liste.

2.3 Gli Score

Molto utili sono gli "Score": Punteggi. Si posono assegnare dei punteggi ai messaggi in modo da meglio identificarli o cancellarli automaticamente. I punteggi sono cumulativi, quindi se

un messaggio rispetta più condizioni ad esso sarà assegnata la somma di tutti i punteggi.
IMHO: Ritengo personalmente che molti abusano troppo dei killfile, così che questi non contengono solo gli indirizzi dei troll (che meritano questo posto) e precauzioni per lo spam. Spesso finiscono nell killfile anche i Subject delle FAQ, così che un poveraccio che, non trovando risposte nelle FAQ, rimane senza il supporto di persone che potrebbero dargli una mano se solo avessero il buonsenso di usare bene un filtro. Morale della favola: Se usate procmail per cestinare la posta, l'avete persa per sempre. Se usate uno scorefile, mutt vi segnala quelli che verranno cestinati, se proprio volete buttarli via basterà dire un "si" [1] all'uscita di Mutt, altrimenti date una "passata" ai messaggi segnati e chissà potreste scoprire che la regola che avete impostato in realtà non funziona bene e vi cestina posta importante ;-)

2.3.1 Marchiamo i messaggi

la prima cosa da fare per applicare uno score è inserire la colonna degli score :-). Questo si fa modificando la seguente riga del `.muttrc`

```
set index_format = "%3C %Z %{%b %d} %-20.20L (%3l) %s"
```

ed aggiungendovi `%3N` [2] dove più vi aggrada. Ad esempio

```
set index_format = "%3C %3N %Z %{%b %d} %-20.20L (%3l) %s"
```

Altra buona consuetudine è quella di riordinare (se volete, ovviamente) i messaggi in base agli score che hanno ricevuto. Quindi cambiate la chiave `set sort_aux` in

```
set sort_aux=score
```

A questo punto non resta altro che assegnare gli score.

In fondo al file di configurazione `.muttrc` c'è la sezione `SCORING`. Uno score segue la seguente struttura

```
score <pattern> <valore>
score <pattern>=<valore>
```

Le due differiscono per una importante finezza. Se l'uguale non è presente, al messaggio viene assegnata la somma di tutti i punteggi a cui corrisponde. Se, invece, l'uguale è presente al messaggio viene assegnata la somma di tutti i punteggi fino all'espressione che contiene l'uguale (compresa). Le altre vengono ignorate. Facciamo un esempio:

```
score <pattern-1> 100
score <pattern-2> 100
score <pattern-3> 100
```

Se il messaggio rispetta le condizioni specificate in tutti e tre i pattern gli viene assegnato il punteggio di 300.

[1] Non preoccupatevi, è un si poco impegnativo :-)

[2] Il 3 serve a dare 3 caratteri di spazio per meglio incolonnare i dati mentre N introduce la colonna degli score

```
score <pattern-1> 100
score <pattern-2>=100
score <pattern-3> 100
```

In questo caso anche se messaggio rispetta le condizioni specificate in tutti e tre i pattern gli viene assegnato il punteggio di 200. Infatti il pattern-3 non viene analizzato a causa dell'uguale nel pattern-2 che causa l'uscita. Spero di essere stato chiaro :-).
Gli score sono utili anche quando, soprattutto nelle mailinglist, vogliamo ritrovare subito i messaggi che abbiamo spedito noi stessi.

```
score '~f michelelug@telug.it' 333
```

Questa regola assegna a tutti i nostri messaggi il numero 333, e se tale numero è sufficientemente alto i nostri messaggi galleggeranno fin su in cima alla lista :-)

2.4 Gestire i messaggi

Questa è forse la sezione più interessante in quanto qui troviamo tutta una serie di "trucchetti" di utilizzo quotidiano.

2.4.1 Gli allegati

Allegare un file

Abbiamo già visto che un allegato si inserisce quando abbiamo già finito di scrivere il messaggio e mutt ci chiede cosa vogliamo fare di quel messaggio. Di solito battiamo y (Spedisci) e il messaggio parte. Prima di spedire il messaggio possiamo anche decidere di non spedirlo (q), cambiare il campo To: (t), cambiare il campo Cc: (c), il Subject: (s) oppure allegarci un file (a). Alla pressione del tasto a mutt ci chiede in modalità " ultima linea" (cioè in basso) di scrivere il percorso relativo a assoluto del file. In alternativa inserendo "?" si apre un più comodo navigatore.

Vedere, stampare e salvare gli allegati

Se invece un messaggio contiene un allegato battendo il tasto "v" si apre una nuova interfaccia che ci permette di gestire gli allegati presenti.

I comandi possibili sono riassunti di seguito.

```
^E              modifica il tipo di allegato
<Return>        visualizza l'allegato usando se
                   necessario la voce di mailcap
^K              estrai le chiavi pubbliche PGP
<Esc>e          usa il messaggio corrente come modello per uno nuovo
L               rispondi alla mailinglist indicata
T               visualizza l'allegato come se fosse testo
b               rispedisci un messaggio a un altro utente
d               cancella la voce corrente
f               inoltra un messaggio con i commenti (quoting)
```

```
g               rispondi a tutti i destinatari
h               visualizza il messaggio e (dis)attiva la rimozione degli header
m               forza la visualizzazione dell'allegato usando mailcap
p               stampa la voce corrente
r               rispondi a un messaggio
s               salva in un file un messaggio/allegato
u               ripristina la voce corrente (undelete)
|               manda un messaggio/allegato a un comando della shell con una pipe
```

Risulta comodo poter cancellare gli allegati attaccati ad un messaggio senza per questo perdere informazioni sull'allegato che ad esso era presente. Ad esempio a me piace raccogliere gli allegati in una cartella Attach. Se un messaggio contiene come allegato il file nomefile.ext, so che il file è presente nella directory Attach.

Sconsiglio di tenere gli allegati attaccati al messaggio. In primo luogo accade spesso che si cerca un file che si ricorda ever ricevuto come allegato ma poi si perdono ore a cercare nella distesa di mailbox il messaggio che lo conteneva. Secondo poi a cosa serve avere 2 copie dello stesso allegato ? Si è vero che un CD-R costa 1 euro ma i masterizzatori non sono ancora a disposizione di tutti.
Comunque sia queste sono regole che ognuno di noi impara in base alla sua esperienza e soprattutto in base alle sue esigenze e disponibilità di mezzi.

2.4.2 Stampare i messaggi

Per stampare i messaggi, nemmeno a dirlo, c'è bisogno di configurare la stampante, ma questo esula dalle finalità di questo HowTo e pertanto ci limiteremo soltanto a riportare lo stretto necessario.

Le opzioni per la stampa sono riportate nella sezione "Editor e Stampa" del `.muttrc` allegato a questo manuale. In ogni caso per definire le opzioni di stampa si devono modificare le voci

```
set print=ask-yes
set print_command=/bin/false
```

che per default hanno i valori che vedete assegnati, cioè si chiede la conferma positiva della stampa [3] e si dice che in realtà la stampante non esiste. Quindi per stampare dobbiamo prima configurare un stampante una stampante. Ad esempio decidiamo di definire una configurazione della stampante col nome di `lp360`[4].

A questo punto possiamo inviare il messaggio alla stampa in molti modi.

Utilizzo di mpage

`mpage` è un programma molto potente e versatile per stampare il file in formato testo in quanto consente la stampa di piú pagine per foglio (utile per salvare la foresta amazzonica) ed inoltre converte il file direttamnente in ps[5]. Per utilizzare mpage bisogna installarlo ;-).

[3] ask-yes significa che la richiesta imposta di default una affermazione. Al contrario ask-no significa che la richiesta imposta di default una negazione

[4] La vostra stampante potrebbe chiamarsi lp, lp0, lp1 etc ... . Io l'ho chiamata lp360 per ricordarmi che richiamando quel device la stampante stampa con una risoluzione di 360dpi

[5] In generale provate a reindirizzare l'output di mpage su un file e vedete cosa ottenete ;-)

Fatta l'installazione basta scrivere nel nostro `.muttrc` la seguente riga nella sezione "Editor e Stampa":

```
set print=ask-yes
set print_command=mpage -2 -bA4 -o -Plp360
```

Abbiamo detto ad `mpage` che deve stampare 2 pagine per foglio, su un foglio A4, sulla stampante lp360 e che deve togliere la cornice ai margini[6].

Il comando stampa

Per stampare un messaggio da mutt basta premere il tasto p mentre si visualizza il messaggio oppure quando si è nella lista dei messaggi e il cursore aforma di freccia è al fianco del messaggio che si desidera mandare in stampa.

2.4.3 I link degli url nel messaggio

Molto spesso ci arrivano nei messaggi informazioni riguardanti siti internet ed in particolare il loro indirizzo (il link appunto). Se si dispone del pacchetto `urlview` è possibile estrarre i link presenti nel messaggio per aprirli con il nostro navigatore preferito. Quindi appurato di aver installato il pacchetto `urlview`, nel momento in cui si visiona il messaggio basta premere i tasti `Ctrl+b` per fare apparire la lista di tutti link presenti nel messaggio. Scorrendo con i tasti freccia e premendo invio si apre il navigatore web che aprirà il sito linkato.

Si noti però che il browser preferito per l'apertura del link deve essere configurato nei file a corredo di `urlview` e non di mutt.

Il file a cui si deve far riferimento è `/usr/bin/ur_lhandler.sh` . In questo file consiglio di scambiare i programmi utilizzati per l'apertura dei siti http e cambiare la riga in questione come la seguente:

```
http_prgs="/usr/bin/netscape:XW /usr/bin/lynx:XT"
```

Altrimenti gli url http (la maggioranza dei siti) verrà aperta con lynx in un xterm anzichè con Netscape (di solito più gradito rispetto a lynx)

Per approfondimenti si faccia riferimento, come al solito, al manuale di urlview.

2.5 Gli "hook": Ad ogni cartella la sua regola

`[[Still Need Work]]`

Mutt non poteva di certo abbandonarci alle nostre stravaganze e così, armatosi degli `hook`, si mette a nostra disposizione definendo per ogni mbox una configurazione distinta. Cosa voglio dire ? Semplice.

Fino ad ora abbiamo definito dei parametri che sono uguali per tutte le cartelle che `procmail` ci ha creato senza fare distizione di sorta. E se volessimo che, ad esempio, la cartella della posta privata sia ordinata per data anzichè per thread ? Basta ridefinire, tramite un `hook`, la variabile appropriata per quella sola cartella. Ma gli hook non si fermano qui. E'

[6] Se si toglie l'opzione -o mpage mette una fastidiosa cornice attorno al teso. Si rimanda alle man pages di mpage per approfondimenti

possibile definire una signature per ogni cartella, una identità diversa per ogni cartella [7], un tipo di header per ogni cartella e via discorrendo. In poche parole ogni variabile che esiste nel `.muttrc` può essere propriamente definita per ogni cartella creata. Questa è parte della flessibilità che offre mutt.

2.5.1 Una identità per ogni Mailing List

Quando ci si iscrive ad una ml si vuole ad esempio usare un nickname diverso oppure si vuole semplicemente che l'indirizzo di reply sia diverso. Ad esempio io preferisco che quando scrivo sulla ml del telug il reply sia l'indirizzo che ho presso il telug, mentre se scrivo sulla ml topomail voglio cambiare del tutto identità e scrivere il replay in maniera criptica in modo da evitare che progammi automatizzati captino il mio messaggio e mi riempino di spam la mia mailbox.

Per fare questo ci si può avvalere della chiave `send-hook`. La sintassi da usare è la seguente:

```
send-hook [!]pattern command
```

Facciamo un esempio. Supponiamo di essere iscritti alla ml telug@telug.it e alla ml linux@topomail.it

```
send-hook telug 'my_hdr From: Michele Antonecchia - <michele@telug.it>;\
                 my_hdr Reply-To: michele@telug.it'
send-hook linux 'my_hdr From: Gamba di Legno; \
                 my_hdr Reply-To: nospam@michele.telug.it'
```

Analizzare cosa ci sia scritto non è molto difficile. Quando scrivo alla ml del telug vengono risettati i campi `From:` e `Reply-To:`. I due "comandi" sono separati da un punto e virgola poi ho aggiunto un backslash per andare a capo per avere una scrittura più pulita. Analogamente per la ml topomail. Ovviamente avrei potuto cambiare qualsiasi altra chiave, oppure cambiarne di più, ma non posso di certo fare tutti gli esempi ;). Invece una cosa mi preme sottolineare. Nella sintassi che ho presentato non compaiono gli apici. Ebbene questo è dovuto al fatto che l'azione di `send-hook` non è un comando vero e proprio ma contiene, in questo caso, una indicazione.

A questo punto nasce però un piccolo problema. Se scrivo una email alla ml del telug e poi ad un amico che ha un indirizzo non contenuto nei `sed-hook`, rimarranno impostate le opzioni per la ml del telug. Cioè il campo `Reply-To:` conterrà `michele@telug.it`. Per risolvere questo inghippo devo fare in modo che per tutte le altre email che scrivo le impostazioni sono quelle che voglio. Quindi la sezione dei `send-hook` diventa la seguente.

```
send-hook .* 'my_hdr From: Michele Antonecchia - <indirizzo di default>;\
              my_hdr Reply-To: indirizzo di default'
send-hook telug 'my_hdr From: Michele Antonecchia - <michele@telug.it>;\
                 my_hdr Reply-To: michele@telug.it'
send-hook linux 'my_hdr From: Gamba di Legno; \
                 my_hdr Reply-To: nospam@michele.telug.it'
```

[7] Ad esempio io preferisco utilizzare lo stesso nickname utilizzato per i newsgruops anche nelle mailing lists.

A questo punto posso andare nella sezione `Header Personalizzati` del `.muttrc` e posso commentare le chiavi `my_hdr From:` e `my_hdr Reply-To:`.

N.B.: Per usare gli header personalizzati deve essere attiva la chiave `set hdrs`.

2.5.2 Ad ogni amico la sua cartella

Se avete una corrispondenza molto fitta con un amico vorrete di sicuro creare una mailbox destinata soltanto a lui. Per quanto riguarda la posta in ingresso non ci sono problemi. Ci penserà Procmail a smistare la posta in ingresso.

```
:0
* ^From.andrea@pippo.it.*
$MAILDIR/Friends/andrea

:0
* ^From.*chiara@paperino.it.*
$MAILDIR/Friends/chiara
```

E fin qui non ci sono problemi. Per quanto riguarda la posta in uscita, invece, dobbiamo designare Mutt come "smistatore". Infatti di default la posta in uscita viene salvata nella mailbox designata dal comando `set record=+mbox`. Se vogliamo che la posta in uscita di Chiara e Andrea venga salvata nelle loro rispettive mailbox assegnate dobbiamo ricorrere agli `hook` di Mutt, ed in particolare a `fcc-hook`. Scriviamo quindi nel `.muttrc`:

```
fcc-hook '~t ^andrea@pippo.it' +Friends/andrea
fcc-hook '~t ^chiara@paperino.it' +Friends/chiara
```

In questo modo la posta "da" e "per" i nostri amici è archiviata in una sola mailbox e possiamo seguire il thread senza troppa fatica.

Purtroppo, al momento, non ho trovato una soluzione migliore e inserire un nuovo amico in una friend-list richiede di editare due file, il `.procmailrc` ed ikl `.muttrc`. Speriamo che in un futuro non molto lontano Mutt riesca a colmare questo gap.

2.6 IMAP

Mutt può interfacciarsi direttamente con il server IMAP senza quindi aver bisogno di usare fetchmail per scaricare la posta. Ma si ricordi: il supporto IMAP è ancora sperimentale (nella versione 1.2.5) e non supporta tutte le feature. Dal README riporto di seguito le feature che sono attualmente in fase di sviluppo:

```
* Tab-completion of IMAP folder names
* Folder browsing
* Go-fast stripes
* Postponed-message support
* Server-side copy
* Fast sync
```

```
* Secure login (GSSAPI and CRAM-MD5)
* Attach messages from IMAP folders
* Use an IMAP path as your Maildir (set folder={...})
* Preserve message keywords
* Preserve deleted messages if you don't choose to expunge them
* Delete mailboxes (from the browser)
* Multiple IMAP usernames
* Read-only folder support (toggling read-only is buggy, though)
* More and better segfaults
```

Per far puntare Mutt ad una mailbox IMAP dobbiamo scrivere la nostra mailbox come

```
{hostname} mailbox
```

Dove `hostname` è il nome del server IMAP, e `mailbox` è il nome della nostra mailbox sul server IMAP. Tutti i server IMAP forniscono una speciale cartella chiamata INBOX, nella quale arrivano tutte le nostre mail. [8] Per esempio, se il nostro server IMAP è `mail.topolug.it` allora dobbiamo aprire la cartella INBOX dicendo a Mutt di aprire

```
{mail.topolug.it} INBOX
```

come se fosse una comune mailbox, cioè premendo il tasto "c".
E qui, insieme a Marco Pratesi, abbiamo scoperto che se siete amministratori e avete installato il server imap della Washington University (la stessa del wuftp) allora avete anche un bel problema: Mutt, e non solo, si trasforma in un bel filemanager da remoto in grado di esplorare tutto il vostro filesystem. Alla faccia della sicurezza. Comunque tutto è documentato su bugtraq. Permettetemi ora una mia personale opinione: ma alla wu sanno programmare ?

Se necessario, si può specificare anche la porta del nostro server IMAP, e chiedere a Mutt di usare `ssl` se disponibile. La sintassi completa a tal proposito è

```
{[user@]hostname[:port][/ssl]} mailbox
```

2.6.1 Connettersi al server IMAP

Per connettersi al server IMAP, bisogna aprire la cartella IMAP. Cioè, premere il tasto `c` per passare al comando "apri cartella" che torna con il prompt `Open mailbox:`, ed introdurre il nome della cartella IMAP come descritto sopra. A questo punto avremo bisogno delle nostre chiavi di accesso `username` e `password`. Possiamo settare queste chiavi direttamente come variabili d'ambiente:

```
imap_user
imap_pass
```

Possiamo settare queste chiavi direttamente nel nostro `.muttrc` (o, preferibilmente in un file con permessi 600 che verrà poi richiamato da `.muttrc` con il comando source), o settarle a mano una volta che Mutt è in esecuzione. La sintassi è la seguente.

```
set imap_user=topolino
set imap_pass=basettoni
```

Se non le abbiamo settate, Mutt ci avviserà quando tenteremo la nostra prima connesione.

[8] Se invece abbiamo creato nuove mailbox sul nostro server, ad esempio perchè abbiamo creato delle regole, allora dobbiamo ricordarci anche di loro

2.6.2 Collegarsi direttamente al server IMAP

Se vogliamo riporre `fetchmail` e `webmail` varie possiamo interfacciarci direttamente con il server IMAP seguendo i due punti di seguito.

- Dire a Mutt di usare la nostra IMAP INBOX come la nostra `$spoolfile`:
  `set spooolfile={mail.topolug.it}INBOX`
- Settare la variabile `$folder` alla root IMAP:
  `set folder={mail.topolug.it}`

2.6.3 Ricompiliamo Mutt per l'utilizzo dell'IMAP

Di solito i pacchetti delle nostre distribuzioni sono già pronti per essere utilizzati con il server IMAP (almeno quelli delli RedHat). In caso contrario ci si può sempre affidare ai sorgenti e ricompilare mutt con il flag `--enable-imap`.

Se avete un sistema basato su rpm e non avete attivo il supporto imap, allora procuratevi il pacchetto src.rpm, installatelo e smanettate dentro lo specfile `.mutt.spec` che si trova (una volta installato il pacchetto dei sorgenti) in `/usr/src/redhat/SPECS`.

Se scorrete il file ad un certo punto vi accorgerete che esiste una chiave `CFLAGS` che contiene tutti i flag per la compilazione standard.

```
CFLAGS="$RPM_OPT_FLAGS" ./prepare --prefix=/usr \
        --with-sharedir=/etc --sysconfdir=/etc \
        --with-docdir=/usr/doc/mutt-%{version} \
        --enable-pop --enable-imap \
```

Come vedete io ho già trovato il flag `--enable-imap` impostato, in caso contrario avrei dovuto aggiungercelo e salvare. [9]

A questo punto resta soltanto costruire il pacchetto rpm. Da root spostatevi nella directory degli specfile, che sulla RedHat è `/usr/src/redhat/SPECS`, e date il comando

```
rpm -bb mutt.spec
```

Il pacchetto appena generato si troverà in `/usr/src/redhat/RPMS/i386` e potrete installarlo come un normale rpm.

Infine se volete ottimizzarlo per la vostra achitettura, basta aggiungere il flag `--target`. Ad esempio per una architettura Intel i686:

```
rpm --target i686 -bb mutt.spec
```

Per la versione 3 di rpm, altrimenti per la versione 4

```
rpm --target=i686 -bb mutt.spec
```

Prima di ricompilare vi consiglio di cambiare il numero di versione del pacchetto. In questo modo saprete in futuro quale pacchetto avete installato, l'originale o quello taroccato da voi. A tal proposito sempre nel `mutt.spec` cambiate, ad esempio, la riga

[9] Vi ricordo che il simbolo \ significa soltanto che il comando continua sulla riga successiva.

```
Version: %{pversion}i
```

nella seguente:

```
Version: %{pversion}i.imap
```

In questo modo il pacchetto rpm generato si chiamerà `mutt-1.2i.imap-2.i686.rpm`. (Sempre che la versione di mutt sia la 1.2i.2). Se, dopo aver installato il pacchetto, date un bel `rpm -qa | grep mutt` capirete a cosa è servito lo sforzo ;)
Questa è la bellezza di Linux ed in particolare di mutt: solo quello che serve.

2.6.4 Utilizzare SSL com IMAP

Compilazione

Se si vuole utilizzare il supporto SSL in mutt, è necessario installare le librerie e gli header di OpenSSL (http://www.openssl.org) prima di ricompilare mutt. Le versioni testate sono le 0.9.3 e la 0.9.4.

Affinchè possa essere abilitato il supoporto SSL, bisogna ricompilare mutt con i flag `--enable-imap --with-ssl[=PFX]`. Se però le librerie e gli header di OpenSSL non sono installate nel percorso di default (di solito `/usr/include` e `/usr/lib`) allora possiamo usare l'opzione PFX per definire la radice della nostra installazione. Le librerie saranno allora cercate in `PFX/lib` e gli header in `PFX/include/openssl`. Quindi, è come se di default PFX fosse `/usr`
Senza ripetermi oltre, se avete un sistema basato su rpm la procedura è la solita: mettere le mani dentro `mutt.spec` alla voce `CFLAGS`. Se invece partite dai sorgenti allora il flag va attivato al momento di lanciare il comando `./configure`.

Utilizzo di SSL

Per accedere ad una cartella IMAP tramite SSL, come già detto, il percorso della mailbox da inserire è

```
mailboxes {localhost/ssl}inbox
mailboxes {localhost:994/ssl}inbox
```

Errori

Se al momento dell'accesso otteniamo un errore circa la mancanza di entropia, significa che mutt non è in grado di trovare una sorgente di numeri casuali in grado di inizializzare SSL. Se ciò accade, dobbiamo generare noi la sorgente e dire a mutt dove cercarla. Di solito mutt serca tale sorgente in `$SslEntropyFile` e in `$RANDFILE`, sempre che tali variabili d'ambiente siano settate, ed in `~/.rnd`, nell'ordine indicato.
Se avete installato OpenSSL 0.9.5 (o maggiori), provate ad installare EGD, Entropy Gathering Daemon (http://www.lothar.com/tech/crypto/). Mutt cercherà di collegarsi al socket di EGD nei seguenti luoghi `$SslEntropyFile`, `$RANDFILE`, `~/.entropy` e `/tmp/entropy`. Se il socket non viene cercato, si deve procedere a creare un file statico come descritto sopra.

Certificazione

Ogni volta che un server viene contattato, la sua certificazione è controllata attraverso un certificato valido. Quando viene incontrato un certificato non valido, ci viene chiesto di verificarlo. Se respingiamo il certificato, la connessione sarà terminata immediatamente. Se accettiamo il certificato allora la connessione verrà stabilita, e possiamo anche salvare il certificato in modo da essere automaticamente riconosciuti dal server alla successiva connessione. Il certificato potrà essere salvato in un file specificato dalla variabile `$certificate_file variable`. Di solito tale chiave è vuota (ovviamente), pertanto dovremo settarla con il valore giusto, ad esempio

```
set certificate\_file=~/Mail/certificates
```

Accorgimenti

Se avete fatto tutto ma non riuscite a collegarvi, allora potrebbe essere che il vostro server IMAP non supporta uno dei protocolli SSL.

Esistono diversi protocolli, TLSv1, SSLv2, e SSLv3. Per verificare quale utilizza il vostro server usate i seguenti comandi:

```
openssl s_client -host <imap server> -port <port> -verify -debug -no_tls1
openssl s_client -host <imap server> -port <port> -verify -debug -no_ssl2
openssl s_client -host <imap server> -port <port> -verify -debug -no_ssl3
```

Potete anche combinare le opzioni per riuscire in un collegamento. Una volta che sapete quali protocolli il vostro server IMAP non supporta potrete comunicarlo a mutt, tramite il solito `.muttrc`, con le chiavi `ssl_use_tlsv1`, `ssl_use_sslv2` e `ssl_use_sslv3`, usando sempre il comando `set` anteposto alle chiavi. Ad esempio

```
set ssl_use_sslv2=yes
```

Usare sempre SSL

Se volete evitare di dichiarare sempre, allora settate la chiave `imap_force_ssl`, sempre con `set` davanti. Ad esempio

```
set imap_force_ssl=yes
```

2.7 POP

Per "par condicio" analizziamo adesso anche il supporto al server POP. In realtà servirsi di questa feature è poco utile in quanto il protocollo POP3 non permette tutte le funzionalità che invece vanta l'IMAP4. Comunque ...
Se mutt è compilato con il supporto POP3 (vedi sezione IMAP con la differenza di usare il flag `--enable-pop`, ovviamente), si è in grado di scaricare la posta connettendosi direttamente al server POP per poi "navigarla" il locale. Quando diciamo a mutt di scaricare la posta dal server POP (default: tasto G), Mutt tenta il collegamento al server specificato dalla chiave `pop_host`, specificata come al solito nel nostro `.muttrc`, e si autentica con la username settata dalla chiave `pop_user`. A connessione stabilita, mutt ci chiede la password (quella relativa alla casella POP in uso, ovviamente).

Una volta autenticati mutt scarica la posta e la piazza nello spool locale. In pratica abbiamo semplicemente fatto fare a mutt quello che avrebbe fatto `fetchmail`. Con la differenza che `fetchmail` è molto più flessibile.
Ne è valsa davvero la pena ?
Boh ... se lo chiede anche il manuale di mutt ;)

Capitolo 3

Patch

In questa sezione vedremo come aggiungere, tramite le patch, nuove funzionalità a mutt. In particolare vedremo come applicare una patch ad un pacchetto rpm. Perchè rpm ? Innanzitutto perchè utilizzo una RedHat e non una Debian, e poi se siete abituati ai tar.gz comunque troverete informazioni interessanti. In futuro ... chissà ;)
Consiglio vivamente di scaricare i sorgenti in formato src.rpm, ma soprattutto che siano adatti alla vostra distribuzione. Se tenterete, ad esempio, di compilare un pacchetto per la RedHat7.x su una RedHat6.x, mutt potrebbe non funzionare correttamente a causa dei percorsi delle directory che non sempre corrispondono.

3.1 Mailbox compresse

Se siete iscritti ad una maillist, avrete notato che mutt dopo poco farà molta fatica ad aprire una mailbox che contiene anche solo 1000 messaggi. Cosa fare allora ? Semplice, archiviare la posta vecchia e comprimerla per non occupare spazio inutile su harddisk. Anche se avete HD da decine di gigabyte vi accorgerete che un traffico intenso non ci mette molto a far lievitare la directory Mail a 100MB.[1]
Ognuno si può inventare il suo metodo di archiviazione: manuale, automatico oppure misto. Io ve ne propongo uno.

Fate capolino alla sezione su `Procmail` e vi accorgerete che `Procmail` è in grado di archiviare in modo del tutto automatico e trasparente la posta in arrivo, ad esempio in file separati per data [2] o in un unico file (metodo consigliato).
A questo punto vi propongo una regola per `Procmail` da aggiungere al vostro `.procmailrc`, di questo tipo:[3]

```
ARCHIVEDIR=$MAILDIR/Archive
:0 c
* ^From.*mailing-list@zio.paperone
| gzip >> $ARCHIVEDIR/zio.paperone.gz

:0
```

[1] Una media di 4000 messaggi occupa circa 15MB. Dopo compressa, la mailbox occupa circa 2.5MB.

[2] Metodo sconsigliato. I replay ad un messaggio potrebbero capitare nel file compresso del mese successivo, rendendo le ricerche faticose

[3] Siate consapevoli di quello che fate. Io non mi assumo nessuna responsabilità ;)

```
* ^From.*mailing-list@zio.paperone
$MAILDIR/zio-paperone
```

In questo modo la prima regola archivia il messaggio senza spostarlo dallo spool, mentre la seconda passa il messaggio in una mailbox non compressa. Quando la mailbox non compressa diventa troppo pesante, potremo buttar via un pochetto di messaggi superflui, tanto ne avremo comunque una copia in archivio che è pure compresso. (ricordatevi di creare la directory `Archive`)
Adesso nasce il problema: " Come leggo la posta nell'archivio compresso?". Ci sono due strade.

- Apro una shell, scompatto làrchivio in un file temporaneo, dico a mutt di pescare la posta in quel file.
- Dono a mutt l'abilità di leggere direttamente le mail compresse

Se vi basta la prima opzione fermatevi qui. Altrimenti andate sul sito di Roland Rosenfeld `http://www.spinnaker.de/mutt/` e scaricatevi la patch relativa alla versione esatta del vostro mutt. Dopodichè installate il pacchetto dei sorgenti.
Adesso inizia lo smanettamento vero e proprio. Copiate il file della patch (ad esempio `patch-1.2.rr.compressed.1`) nella directory `/usr/src/redhat/SOURCES/` e rinominatelo `mutt-compressed.patch`. Dopodichè spostatevi in `/usr/src/redhat/SPECS/` ed editate il file `mutt.spec`.
Per maggiore chiarezza vi ripoto uno scorcio del mio `mutt.spec`.

```
...
Copyright: GPL
Group: Applications/Internet
Source: ftp://ftp.mutt.org/pub/mutt/mutt-%{pversion}i.tar.gz
Patch0: mutt-nosetgid.patch
Patch1: mutt-default.patch
Url: http://www.mutt.org/
...
%prep
%setup -n mutt-%{pversion}
%patch0 -p1 -b .nosetgid
%patch1 -p1 -b .default
...
CFLAGS="$RPM_OPT_FLAGS" ./prepare --prefix=/usr \
        --with-sharedir=/etc --sysconfdir=/etc \
        --with-docdir=/usr/doc/mutt-%{version} \
        --enable-pop --enable-imap \
...
```

Come potrete facilmente immaginare nella prima sezione devo aggiungere la dichiarazione della patch per le mailbox compresse. Nella sezione successiva, analogamente devo aggiungere la patch vera e propria ed in fine devo aggiungere il flag `--enable-compressed` affichè mutt possa arricchirsi delle funzionalità offerte dalla patch. Quindi per farla breve il `mutt.spec` diventa:

```
...
Copyright: GPL
Group: Applications/Internet
Source: ftp://ftp.mutt.org/pub/mutt/mutt-%{pversion}i.tar.gz
Patch0: mutt-nosetgid.patch
Patch1: mutt-default.patch
Patch2: mutt-compressed.patch
Url: http://www.mutt.org/
...
%prep
%setup -n mutt-%{pversion}
%patch0 -p1 -b .nosetgid
%patch1 -p1 -b .default
%patch2 -p1 -b .compressed
...
CFLAGS="$RPM_OPT_FLAGS" ./prepare --prefix=/usr \
        --with-sharedir=/etc --sysconfdir=/etc \
        --with-docdir=/usr/doc/mutt-%{version} \
        --enable-pop --enable-imap \
        --enable-compressed \
...
```

Ci sarebbe molto da dire su come applicare una patch, ma questo non è un manuale sul rpm. Ovviamente se il vostro `mutt.spec` contiene più o meno patch, la patch che inserite voi avrà sempre l'ultimo numero disponibile.
Per maggiore pulizia e per una nota per il futuro ci sarebbero altre modifiche da fare: aggiungere nei `changelog` la modifica che avete fatto e cambiare il nome al pacchetto rpm che verrà generato.

Riporto di seguito queste modifiche:

```
...
%define pversion 1.2
Version: %{pversion}i.comp
...
%changelog
* Wed Mar 07 2001 Michele Antonecchia <michele@telug.it>
- added the compressed folder patch (by Roland)
- added --enable-compressed flag
...
```

Detto questo non resta altro che compilare il pacchetto e installarlo. Quindi da root eseguite, sempre restando nella dir `/usr/src/redhat/SPECS/`, il comando `rpm --target i686 -bb mutt.spec` per rpm versione 3 o il comando `rpm --target=i686 -bb mutt.spec` per la version 4 [4] ed installate il pacchetto che avete generato e che si troverà in `/usr/src/redhat/RPMS/i686`.
Infine c'è un'ultima modifica da fare. Non sono riuscito a reperire la patch e per questo dovremo operare a manina sul file `/etc/Muttrc` aggiungendo in coda le seguenti righe

[4] Se avete un Pentium II o superiore, ovviamente, altrimenti sostituite a i686 la sigla adatta al vostro processore

```
# Compressed Folder Support
#
# gzip
open-hook \\.gz$ "gzip -cd %f > %t"
close-hook \\.gz$ "gzip -c %t > %f"
append-hook \\.gz$ "gzip -c %t >> %f"
#
# bzip2
open-hook \\.bz2$ "bzip2 -cd %f > %t"
close-hook \\.bz2$ "bzip2 -c %t > %f"
append-hook \\.bz2$ "bzip2 -c %t >> %f"
```

Che diranno a mutt come comportarsi difronte ad una mailbox compressa.

Capitolo 4

Tips & Tricks

In questa sezione vengono raccolti alcuni espedienti per arricchire Mutt di particolari feature oppure per risolvere particolari esigenze.

4.1 All'Ufficio e in Facoltà quando non si è root

In molte occasioni capita che si ha la connessione ad internet ma che non ci si possa avvalere dell'MTA installato, come accade nei laboratori dell'Università oppure in ufficio, e non si vuole usare il classico WebMail. In questo paragrafo spiegheremo come "bypassare" l'MTA di default senza per questo contravvenire alle regole che ci ha dettato il nostro Amministratore. Infatti quello che faremo non va oltre ad un semplice accesso sulla porta 25 del nostro server smtp.

NOTA: Per verificare che non siete coperti da un firewall, prima di procedere oltre e perdere tempo fate un telnet sulla porta 25 del vostro server smtp. Se non potete raggiungere il server non riuscirete mai a spedire un messaggio :(

NOTA: Dovete essere in grado di accedere ad un server SMTP in grado di fare da relayer, altrimenti non riuscirete mai a mandare una email. Ad esempio Tiscali, Libero, Tinit etc . . . spediscono messaggi all'interno del loro dominio solo se siete collegati ad internet tramite un loro account, ovvero fate parte del loro dominio

Il programma a cui faremo riferimento è `ssmtp-2.39`, facilmente rintracciabile su `http://www.google.it` . Non è l'unico ma in questa sede non possiamo considerare tutti gli MTA che esistono.

In realtà `ssmtp` nasce come un piccolo client SMTP da sostituire ai colossi `sendmail` ,`postfix` etc. . . su piccole postazioni home. Per questo dovremo fare qualche piccola correzione prima di installarlo.

Innanzitutto scarichiamo il tarball e lo scompattiamo dove ci pare. Poi editiano il file `configure` ed impostiamo le variabili `exec_prefix` e `prefix` come segue:

```
exec_prefix=/home/tuo-utente/ssmtp
prefix=/home/tuo-utente/ssmtp
```

A questo punto creiamo il `Makefile` eseguendo il `configure`:

```
./configure
```

Creato il file `Makefile` lo editiamo e modifichiamo la variabile `etcdir` da

```
etcdir=/etc
```

in

```
etcdir=/home/tuo-utente/ssmtp/etc
```

In pratica abbiamo scambiato la radice principale con la nostra home directory in modo da non cercare di scrivere dove non ci è permesso.
Proseguiamo quindi con la compilazione e l'installazione

```
make ; make install
```

Durante l'installazione ci verranno chieste alcune cose come:

```
Mail name [localhost.localdomain]:
```

a cui possiamo rispondere con fantasia o con un nome di dominio vero, e

```
Please enter the SMTP port number [25]:
```

a cui possiamo confermare, nella maggior parte dei casi, il 25.
Non abbiamo finito :)

Se torniamo nella nostra home directory vedremo che è stata creata la dir `ssmtp` con la struttura seguente:

```
ssmtp/
|-- etc
|   `-- ssmtp
|       |-- revaliases
|       `-- ssmtp.conf  (file di configurazione)
|-- man
|   `-- man8
|       `-- ssmtp.8     (pagina man)
`-- sbin
    `-- ssmtp           (eseguibile)
```

Il tutto per la modica cifra di **64k** !
Bene, editiamo il file `ssmtp.conf` in cui cambieremo la variabile `mailhub` in

```
mailhub=smtp.nostroprivider.it
```

e se vogliamo attiviamo la variabile `rewriteDomain`.
A questo punto il grosso è fatto. Basta ora dire a Mutt di non usare `sendmail` o `postfix` o quello che vi pare . . .) ma il nostro snello `ssmtp`. Quindi editiamo il solito file `.muttrc` e modifichiamo il comando `set sendmail` in:

```
set sendmail="~/ssmtp/sbin/ssmtp"
```

Se il vostro provider o il server che usate per spedire la posta necessitano una autenticazione allora:

```
set sendmail="~/ssmtp/sbn/ssmtp -au user -ap password"
```

Attenti però perchè avete messo la password di accesso al server in un file critico. Comunque il nostro scopo è raggiunto e se vi servono maggiori informazioni su `ssmtp` consultate la pagina di manuale:

```
less /home/sturm/ssmtp/man/man8/ssmtp.8
```

Non fate incavolare il vostro SysAdmin e in bocca al Lupo :)

4.2 Una firma che fa ridere

Quando non ho nulla da fare mi diverto spesso a zuzzurellare per il control center di Gnome. Un giorno scoprii che esisteva un simpatico screensaver con un buffo omino nasuto che, vagando per lo schermo, ne raccontava di cotte e di crude. Incuriosito dal fatto mi sono messo in cerca subito del file in cui l'omino nasuto raccoglieva le sue battute. In quattro e quattr'otto ho scoperto che esiste un simpatico giochino: Fortune. (se non lo avete installatelo ;) Allo stesso tempo mi è venuto in mente che i miei corrispondenti sarebbero stati felici se nelle mail che gli spedivo ci avrei messo anche una barzelletta, o quantomeno così li avrei costretti a leggerla con tale scusa ;)
Esistono vari modi per mettere una "fortune" in una signature. Qui se ne propone uno.
Editate il vostro `.bashrc` ed iserite, nell'aposita sezione, questa riga:

```
alias mutt='creasign ; mutt'
```

A questo punto si deve creare il comando `creasign`, il quale genera la signature nuova.
Create il file `creasign` nella vostra directory `bin`.

```
$ vim bin/creasign
```

[1] e ci scrivete dentro queste righe:[2]

```
#! /bin/sh
#------ Se avete sig_dashes=no ---------
echo "Ciao, Michele" > ~/.signature.mutt
echo "-- " >> ~/.signature.mutt
fortune >> ~/.signature.mutt

#------ Altrimenti se sig_dashes=yes----
fortune > ~/.signature.mutt
```

Mi raccomando. Scegliete SOLO la parte che vi interessa, in base a come la chiave `sig_dshes` è settata nel vostro `.muttrc`.
A questo punto rimane solo una cosa da fare: Permettere a `creasign` di poter creare la signature. Per fare questo eseguite dalla vostra home directory:

```
$ chmod 644 .signature.mutt
```

[1] Attenzione: la vostra directory bin ! Non quella di sistema `bin`
[2] Una letta al paragrafo `La firma` a pagina 7 chiarirà dei dubbi

NOTA 1: A volte `fortune` estrae delle barzellette sconce e/o offensive. In questo caso lanciate il comando

```
fortune -f
```

vi comparirà qualcosa come

```
$ fortune -f
100.00% it
    87.03% italia
    12.97% zozzital
$
```

E' facile capire cosa dice questo messaggio. Pertanto se non volete mettere le cose "zozze" come firma sostituite la parola `fortune` nel file `bin/creasign` come segue:

```
#! /bin/sh
echo "Ciao, Michele" > ~/.signature.mutt
echo "-- " >> ~/.signature.mutt
fortune 100% italia 0% zozzital >> ~/.signature.mutt
```

Mi raccomando non fate i furbi e non scambiate gli indici ;)
NOTA 2: C'è un altro intoppo. Fortune a volte estrae barzellette toppo lunghe, più lunghe dello stesso testo che si manda.
In tal caso fareste bene ad appendere al comando fortune anche i flag -n e -s. Ossia (fate riferimento a `man fortune` ;)

```
#! /bin/sh
echo "Ciao, Michele" > ~/.signature.mutt
echo "-- " >> ~/.signature.mutt
fortune -n300 -s  100% italia 0% zozzital >> ~/.signature.mutt
```

Lascio a voi il buonsenso.
NOTA 3: La stessa procedura può essere ripetuta per slrn [3] o per qualsiasi altro programma. Basta mettere le mani a `bashrc`.

4.3 Aumentiamo la sicurezza: SUDO

Abbiamo visto che non sempre il nostro utente può scaricare la posta perchè non abbiamo i privilegi per utilizzare il comando `sendmail` [4].

`sudo` è un programma alquanto affascinante in quanto limita i pericoli accennati sopra. Infatti creare un link a `sendmail` o assegnarli il bit SUID fa in modo che un utente normale possa usare `sendmail` con tutte le sue potenzialità. Con `sudo` invece possiamo limitare le libertà dell'utente all'uso del programma descrivendo precise regole. Non mi dilungo su `sudo` perchè non è lo scopo di questo manuale, pertanto mi limito a riportare la riga da inserire in `/etc/sudoers` :

```
michele localhost = NOPASSWD: /usr/sbin/sendmail -q, <altro camando>
```

[3] slrn è un ottimo lettore di news. Provalo.
[4] Sia che usiamo `sendmail` sia che usiamo `Postfix` il comando è sempre lo stesso

Già con questo piccolo esempio possiamo apprezzare le potenzialità di sudo. Infatti solo dalla macchina `localhost` e solo l'utente `michele` [5] può eseguire il comando `sendmail -q` senza inserire la password. Badate bene, `michele` non può eseguire `sendmail` se non con il parametro `-q`, qualsiasi altra opzione sarà negata.

A questo punto, però, per spedire la posta si deve eseguire il comando

```
$ sudo /usr/sbin/sendmail -q
```

che se sembra lungo può sempre essere inserito in uno script o in un alias dentro .bashrc .

4.4 Elimina quoting

Spesso, troppo spesso, la gente quota male.
Quotare significa riportare parte del messaggio ricevuto in quello che si sta scrivendo per facilitare la comprensione del discorso.
E' uso comune non riportare più di 4-6 righe o addirittura non più di quanto si scrive nella risposta. Ma a chi lo dici ! La gente fa come gli pare e visto che siamo in democrazia se il mio interlocutore fa come gli pare ... lo castigo con mutt ;)
Quindi se il quoting è troppo e volete eliminarlo, mentre leggete il messaggio, premete il tasto "T" ossia "shift+t".
Per ripristinare il quoting l'operazione è la stessa.

4.5 Una marcia in più (solo rpm)

Smanettando i sorgenti di mutt per scrivere questo manuale, ho notato che si ha un leggero incremento di prestazioni se ricompilate mutt specificando la vostra architettura. Non aspettatevi che mutt metta il turbo o che chissà cosa riesca a fare se ottimizzate il pacchetto. Otterrete solo un leggero miglioramento che però si vede.

Quindi scaricatevi il pacchetto src.rpm dei sorgenti ed installatelo. Da root andate in `/usr/src/redhat/SPECS` e date il comando

```
rpm --target i686 -bb mutt.spec
```

se per esempio avete un pentium II o superiore. Attendete la fine della compilazione, andate in `/usr/src/redhat/RPMS/i686` e reinstallate il pacchetto che ci trovate. Poi fatemi sapere se ne è valsa la pena.

[5] Ossia quello che scarica la posta

Capitolo 5

Introduzione

No! Non mi sono sbagliato. Questa è proprio l'introduzione. Siccome il mio intento era quello di creare un How-To che fosse al contempo completo ma rapido per chi avesse avuto fretta, mi sono reso conto che un "papiello" di chiacchiere all'inizio del documento erano davvero fastidiose. Resta comunque il fatto che qualche parola sulla licenza e sulle finalità del manuale si devono spendere. Quindi ecco a voi volenterosi di tanta lettura tutto quello che c'è da sapere su questo mio lavoro.

5.1 Copyright

5.2 Finalità di questo manuale

Questa opera nasce da quelle che sono state le mie particolari esigenze per configurare mutt.

In questo manuale si partirà dapprima da una configurazione minimale che riguarda soltanto il file .muttrc, in secondo momento, a mò di moduli, si inseriranno delle sezioni che vanno ad arricchire la configurazione minimale. In tal modo l'utente che utilizza, ed esempio, una configurazione del tipo sendmail + mutt + fetchmail + vim, seguirà la sua configurazione ricercando i relativi "moduli" che gli servono e se un giorno deciderà di cambiare qualcosa gli basterà spulciare soltanto la sezione che gli interessa.

Questa opera nasce soprattutto dalla lettura di riviste, HowTo, file di documentazione e quant'altro esiste in rete, edicola e libreria. Il suo intento non vuole essere affatto denigratorio

nei confronti del lavoro di chi ha precedentemete scritto qualcosa in riguardo al tema. L'opera è open, e pertanto affatto con fini di lucro.

Non verrà spiegato il procedimento di installazione dei pacchetti in quanto si ritiene l'argomento del tutto fuori tema, nonchè banale.

Si consiglia, comunque, come approfondimento, di leggere il file di manuale allegato allo stesso mutt che si trova, a seconda delle distribuzioni, nella directory `/usr/doc/mutt-xxx/manual.txt` oppure `/usr/share/doc/mutt-xxx/manual.txt` dove ovviamente xxx indica la versione del programma.

5.3 Note alla versione

Purtroppo la versione che state leggendo è ancora in fase di sviluppo e molti argomenti non sono completi. Mi dispiace per questo inconveniente ma cerco di fare il possibile. L'obiettivo di questo manuale è quello di diventare completo in ogni sua parte. Quando sarà raggiunta la versione 1.0.0 le successive saranno soltanto di revisione e le versioni seguiranno di pari passo quelle di mutt stesso in modo da avere una rapida informazione in riguardo a quale versione di mutt il manuale fa riferimento.

Con la speranza che il mio lavoro sia di aiuto a chiunque utilizza mutt auguro buona lettura e buona configurazione.

5.4 Differenze dalle versioni precedenti

Versione 0.0.8 (=0.1.0rc) (28 settembre 2001)

- Gestire una mailbox per ogni amico
- Usare un MTA alternativo quando non si ha accesso al principale

Versione 0.0.7 (19 luglio 2001)

- Aggiunta la sezione sulle identità
- Cambiato il logo
- Aggiunti i ringraziamenti
- Staccato un biglietto per le vacanze ;)

Versione 0.0.6 (22 maggio 2001)

- Estesa la sezione IMAP: supporto SSL
- Corretta la sezione IMAP
- Inserito il capitolo sulle patch
- Inserita la sezione sulle mailbox compresse
- Aggiunta la sezione sulla sicurezza (SUDO)

- Correzioni varie
- Rivisto e corretto il capitolo 1
- Insrerita una Tips
- Rivista la sezione di fetcmail

Versione 0.0.5 (7 marzo 2001)

- Introdotta la sezione sul supporto IMAP
- Introdotta la sezione sul supporto POP
- Introdotta la sezione sullo "scoring" dei messaggi
- Introdotta la sezione "Una marcia in più" nella sezione T&T
- Inserita la data corretta di pubblicazione (forse)
- Aggiunti altri errori di battitura ;)

Versione 0.0.4 (10 febbraio 2001 e non come pubblicato)

- Introdotto il capitolo 4: Tips & Tricks
- Introdotto il paragrafo "Una firma che fa ridere" nel capitolo 4.
- Spostata l'introduzione alla fine del documento

Versione 0.0.3

- Riorganizzazione dei capitoli, sezioni e paragrafi.
- Rivisitato il file `.muttrc` per l'utilizzo di procmail e delle mailbox.
- Introduzione di un nuovo capitolo per la configurazione approfondita
- Introdotta la sezione di stampa nella gestione dei messaggi
- Introdotta la sezione per gli url presenti in un messaggio
- Correzione di alcuni errori di battitura.
- ... e molte altre cosucce :-)

5.5 Ringraziamenti

Un ringraziamento a Marco Pratesi per avermi concesso lo spazio sul Telug per pubblicare questo lavoro e per avermi passato il suo programma inputtex.c senza il quale avrei preso a calci la RedHat7.1. Inoltre un ringraziamento collettivo a tutti coloro che hanno letto il Mutt-HowTo e che tramite le loro osservazioni hanno contribuito a ritoccarlo. Infine ringrazio in anticipo tutti coloro che esprimeranno, tramite email, un giudizio su tale opera.

Grazie a tutti, Michele.