The Woodnotes Guide to the Mutt Email Client

Randall Wood (www.therandymon.com)

December 2, 2009

Contents

1 Introduction – Why Mutt?

Writing, reading, and sending email is most satisfying when the tools you use facilitate your work. And Mutt will help you do just that. Mutt is a console mail user agent (MUA). It is not graphical, and makes no use of the mouse. Rather, it runs from the command prompt, so you can use it on consoles where the X Window environment is not running, in virtual terminals, and in remote shell sessions. It has no buttons, sliders, or icons: it's text based. Mutt doesn't do some things you might be used to, but it does several things you never knew were possible and it does them well. It's lean and mean, and once you learn how to make mutt work for you it's hard to use anything else without feeling like the software impedes your efficiency.

Mutt is efficient and highly-configurable (like most Unix/Linux software, it has more configuration options than you'll probably ever use). One mutt user can sit down at another user's computer and be unable to do anything useful at all – it's that configurable. That means this Woodnotes guide will get you started, but you will quickly modify the keystrokes and commands to suit your own environment and make this guide obsolete. No offense taken!

Mutt is not the tool of choice for everybody, but if you prefer the keyboard to the mouse, appreciate simplicity but not at the expense of power, and are particular enough about your work environment to choose tools that provide maximum configurability, learning how to use Mutt will make email a lot more fun to deal with.

Top reasons to use mutt over another mail client:

1. it's fast and efficient, and lets you keep your hands on the keyboard, where they belong.
2. mutt makes maximum use out of your limited screen real estate
3. you can use it with the editor of your choice (emacs, vi, nedit, pico, etc.)
4. you know exactly what you like and therefore appreciate mutt's unsurpassed configurability
5. you want fast keystroke access to functionality and therefore appreciate mutt's powerful macro capability
6. you grew up in the stone age, like me, and prefer distraction-free, text-based computing

Great tricks to do with Mutt:

1. cut to the chase when receiving a message inside a message inside a message (sec. 2.4)
2. automatically fold Word doc attachments right into the text of the message (sec. 3.10)
3. archive hundreds of messages at once in one simple step (sec. 2.8.4)
4. make annoying HTML email look like plain text (sec. 3.8)
5. on a dial-up connection, queue up your mail in a sendmail spool file, which is automatically sent the moment an internet connection is detected without any further interaction needed (sec. 3.4)
6. write powerful macros to make complex tasks into a single keystroke (sec. 4.4)
7. delete just attachments from a message without deleting the whole message (sec. 2.4.2)
8. color code messages in your inbox to indicate their origin, or the address the messages were sent to, and more (sec. 4.2)

2 Usage: The Basics

2.1 Getting Help

First of all, in any screen you may press the ? key to be taken to a help menu listing the current key bindings relative to the task at hand. You can search through this help screen for specific topics of interest to you the way you would when using the vi editor or less: hit the / key to enter your search term, and n and p to cycle through the next and previous occurences of that term throughout the document. Type q to exit the help screen. Once you know how to get help, you'll be more tempted to explore, and you should. If your version of mutt has been properly configured, you can type F1 to get to a help file (the man page, actually),

but if not, you can type “man mutt” or “man muttrc” at a terminal prompt to read the man pages. Other help resources are listed in section 5 of this Woodnotes guide.

Assuming little gnomes and trolls have already magically configured your system for you, you can open a virtual terminal window and type ‘mutt’ to proceed directly to reading your mail; otherwise, skim the next sections and proceed to section 3 to read about configuring your system for mutt.

2.2 Reading Messages

2.2.1 Your Inbox (Index View)

As you launch mutt the program’s initial view is of the message index, presenting a list of all messages currently in your inbox. Mutt does the best it can with column widths given the size of your screen or virtual terminal, showing you as much as it can of the author and subject fields.

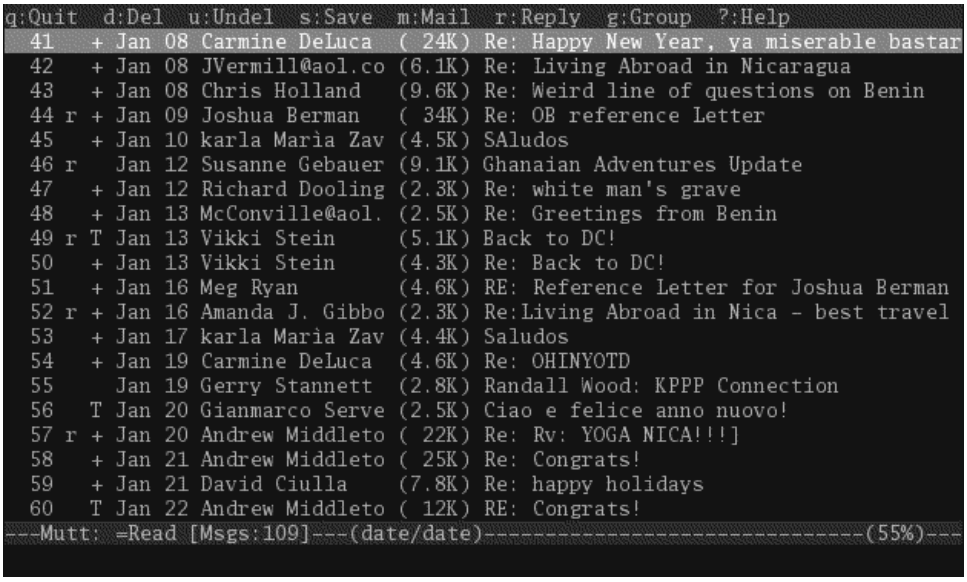

Figure 1: Mutt’s Index View

The first column shows you the message number, and you can navigate to messages just by typing the number. The second, third, and fourth columns show you the status flags associated with each message, such as ‘N’ for new mail, ‘T’ for messages in which you are on the ‘To’ line, ‘C’ where you’re copied, and so on. See figure 2 below for a complete list.

Use the arrow keys or ‘j’ and ‘k’ to navigate up and down through the messages, or type the number of the message and press enter to jump straight to it. When the cursor bar is highlighting a message, press return to read it (that is, go to the ‘pager’ view). When you’ve finished reading it, press ‘q’ to return to the index view. As the menu bar at the top of the screen indicates, you can press ‘f’ to forward a message, ‘r’ to reply, and ‘m’ to compose a new message.

Flag	Meaning
N	new message
O	old message
r	message to which you've replied
D	message marked for deletion
d	message with attachments marked for deletion
T	message where you are on the 'to' line
F	message from you
C	message where you are on the 'cc' line
+	message sent only to you
L	message sent to a subscribed mailing list
⋆	message which has been tagged (see section 2.8.2)
s	message with a PGP signature

Figure 2: Message Flags

2.2.2 Sorting and Threading Messages in the Index View

By default, the messages in your index view are shown in date order, but you can sort on any number of criteria, including date, sender, recipient, message size, and subject, among others. To sort messages, press 'o' and select your criterion. A capital letter indicates reverse sort, so if you press 'z' for size, the messages will be presented in size order from small to large, but if you press 'Z' they will be sorted from large to small.

The most useful view, in my opinion, is the threaded view (press 't' or 'T'), which groups messages into conversations and manages to keep track even when message subjects change through the course of a conversation. Mutt's display of threaded messages is unrivalled in other software and I miss it everywhere else.[1] A bit later, we'll look at limiting the messages you see in this view, which is another powerful way of looking at some of the messages in your inbox (see section 2.8.3).

2.2.3 Reading Messages (Pager View)

From the index view, pressing return with the cursor on any message takes you to the pager view for reading that message. There, mutt provides some functions in the pager that can be extremely useful.

While reading a message, press 'h' to toggle the Header view, in which all the message's headers are shown, and 'h' again to toggle it back to the view without headers. This is useful when you would like to know something about along which route your message traveled, or which program your correspondent used to compose a message, or (these days) what caused a message to be flagged as spam.

You can also press 'T' to hide all quoted text from the view. This only works with messages whose quoted portions are preceded by > signs, but in those cases it's quite useful. It's a toggle, so press 't' again to include the quoted text again.

While in the pager view, mutt might report that the message arrived with attachments. You can press 'v' to see a folder tree of the attachments, their text labels, and sizes. See section 2.4 for dealing with attachments.

[1] I'm told the algorithm that does this is more powerful in mutt than in other software which simply sorts by subject line and date.

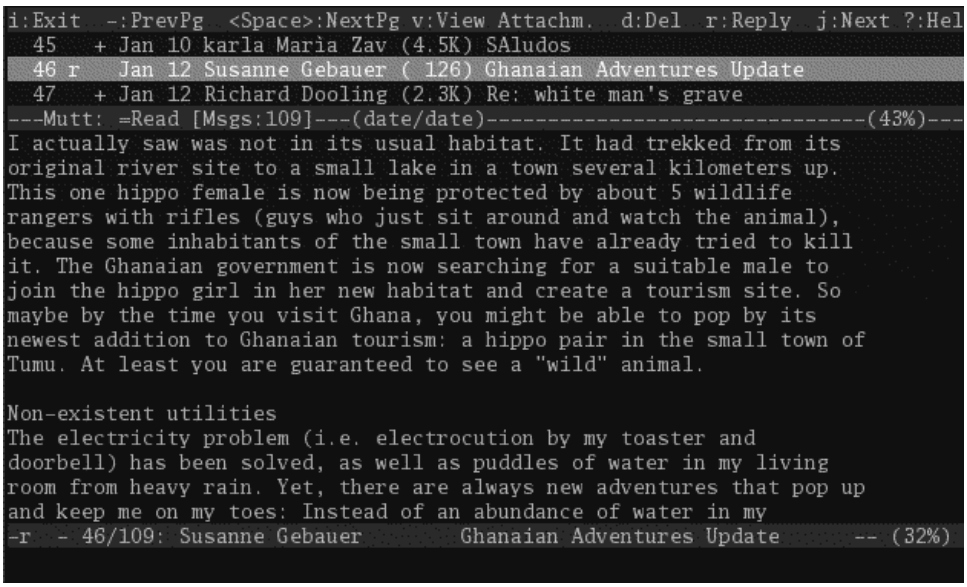

Figure 3: Mutt's Pager View

2.2.4 Deleting Messages

When you press 'd' in the index or pager views to delete a message mutt will flag it with a 'D' in the index view (it may also reflect the deletion using color: my mutt is configured to show deleted messages in red. We'll look at the options available to you in sec. 4.2) – but doesn't actually delete the message until you either change to a new folder or sync the mailbox to commit the changes. To make the deletion permanent, press the $ key, which is the default command to sync the index with the server and actually deletes all messages flagged for deletion. Up until that moment, you still have the option to undelete those messages.

Undelete a message by putting the cursor over the desired message and pressing the 'u' key. You can navigate there either by using the message number, or by holding down shift as you type 'J' and 'K.' Otherwise, the cursor will cavalierly jump right over all deleted messages as you navigate.

2.3 Composing Messages and Saving Drafts (Postponing Messages)

Dealing with individual messages is straightforward. The basic commands for dealing with email messages are printed in the menu bar at the top of your console window, and are shown in table 4 below.

These commands are available to you either from the index (when you can see all of the contents of your mailbox) or the pager (when you're reading a particular message).

Once you press a key to reply to, forward, or compose a message, mutt launches your favorite text editor so you can compose your text[2] The text message is created by default in a temporary directory (with a name probably like /tmp/mutt-2502.txt), and you can save the message as you go along. If Mutt ever crashes,

[2]Mutt adheres more strongly than most programs to the Unix philosophy of "do one thing only but do it well," and rather than

Key	Command
d	delete message
r	reply to message
f	forward message
s	save message
g	(“group”) reply to all
m	(“mail”) compose new message

Figure 4: Mail Commands (Reply, Forward, Etc.)

you can often search for the temporary text files in the /tmp directory. When you’re ready, save the file one last time and exit your editor. Mutt will take you back to the screen where you can add or modify recipient addresses to the address fields, if you haven’t already done so. Follow the prompts at the top of the screen (‘t’ for the To line, ‘c’ for the ‘Cc:’ line, and so on), separating addresses with a comma. If you would like to attach some file to your message, you can do so from this same screen by pressing ‘a.’ When you have finished writing your message and adding your attachments, pressing ‘y’ sends the message.

If you decide at the last minute to cancel the composition of your message and return to it later, instead of pressing ‘y’ at this point, press ‘q.’ Mutt will ask you if you’d like to postpone or kill the current message. Postponing it puts the message into a queue of postponed messages saved locally. When you are ready to continue, pressing Control-r will bring up the message and open your editor to begin working on it, or in the case you’ve got several postponed messages, will bring up an index of them so you can choose which one to continue.

Mutt is surprisingly capable of performing much more complicated tasks at this point. For starters, you can press “i” to spellcheck your message with Ispell (if you haven’t already done so using your text editor), and by pressing “e” with the cursor over the temp file, you can re-edit your message. You can rename attachments, and much more. For a taste of what else is possible, press the ? key (press q to exit the help menu when you have finished).

2.4 Attachments

Either from the index or from the pager view, pressing ‘v’ will take you to a list of all attachments of the message. This might be more informative than you think! For example, many messages come with both plain text and HTML parts (see Figure 5, showing a message sent in both ASCII and HTML and including a picture). You can also take advantage of this to drill down to the heart of messages that have been forwarded and re-forwarded through many senders. Open up the list of attachments, and note which one has the greatest size; put the cursor over it and press enter to read the joke ten people thought so funny they had to forward it to all of their friends. Isn’t that easier than mail software that makes you double click on each of the ten messages in turn?

incorporate its own text editor relegates the task to programs better suited to it. As such, because you are guaranteed of being able to use the text editor you like best to write your message. If you haven’t specified an editor in your configuration file, the system default – probably vi or vim – will launch. We’ll cover this configuration in section 3.

Figure 5: Mutt's Display of the Different Parts of an Email Including Attachments

2.4.1 Forwarding Attachments

If you forward a message, odds are the text of the message will not include any attached files, unless you specify so. This is important. If someone sends you, say a picture, and you would like to forward it onto someone else, you must specify the attachment for forwarding. From this menu, put the cursor next to anything you'd like to forward and tag it (press 't'). Then press ';f' (note the semicolon) to forward everything that's been tagged. While at first this might seem like a hassle, once you get used to it you will appreciate the efficiency. For bonus points, assuming you too want to forward the message to someone else, clean it up by deleting all those other parts of the message so the next recipient doesn't have to dig. For details, continue reading the next section.

2.4.2 Deleting and Undeleting Attachments

It's often convenient to remove an attachment from your inbox but not the message itself. To delete an attachment from a message without deleting the message itself, open a message and then press 'v' to see the list of attachments. Put the cursor over the one you'd like to delete and press 'd.' When you return to the index, the index will show a lowercase 'd' next to the message to indicate the change, and when you sync the mailbox by pressing $, the change will be committed (and the index will reflect the changed size of the message). This is a good way of reducing the size of mail in your mailbox without having to delete any message text, and once you get used to being able to do this, you'll miss it in other software.

2.5 Printing Messages

Mutt hails from the line printer days, so on a Linux or Mac box, the CUPS print server or equivalent should be ready to accept messages sent to the LPR command. You should be able to just press the 'p' key to print a message, assuming your line printer has been set up. The text printed lacks any sort of embellishment, which is fine for most but annoying to some. You can have additional printing functionality by installing the mutt-print package for your distribution. The mutt-print package uses the LaTeX software on your Linux system to make printed email look gorgeous (see sec. 4.9).

2.6 Reading Mail in Other Folders

No one lives out of just their inbox anymore; email users use dozens of folders to separate and store mail of one form or another. Use the 'c' command to change folders, that is to navigate to a different mail folder, and 's' to save the current message to a different folder. There are only two tricks you need to remember:

first, '!' refers to your spool file, or inbox, and second, the '=' key refers to any non-inbox folder. So to change to the 'Work' folder, enter 'c=Work' and hit return. To return to the inbox, hit 'c!' and hit return. Hitting 'c' alone makes mutt prompt you to see a list of folders. At the prompt, hit the ? key to be shown a list of folders. Arrow down through them and hit return to change folders. At just about any time, mutt will try to complete words for you, so you might be able to get away with hitting simply 'c=W' and then pressing tab. If you have no other folders that begin with W the word 'Work' will be completed for you.

Likewise, the 's' key saves the current message to a folder of your choosing. From the index, select the message and hit 's=Work' to save the current message to the folder called Work. If the folder doesn't already exist mutt will query you if you'd like to create it, and will probably suggest a folder name based on that message's sender's name unless configured otherwise. Below is a list of the key mail boxes.

Key	Location
!	Inbox/Spool
+ or =	folder collection (archives)
<	sent mail folder (e.g. =FamilyMail)
~	home directory on local (not remote) system

Figure 6: Special Mailboxes

Special case: IMAP Shared Folders: If you are accessing an IMAP server, there may be shared folders. Exhaustive experimentation reveals that you can access a shared IMAP folder called `Users/Shared/FunStuff` by entering `c imap://mail.server.com/Users.Shared.FunStuff`. That's a lot of typing, so save yourself some trouble and bind that expression to a keystroke (see section 4.3).

2.7 Saving Messages to Files

Let's say you want to convert an email message to a text file saved somewhere on your computer. You can do so while retaining the message in your mailbox, or convert it to a text file and remove it from your mail.

You can copy it to a text file without removing it from your mailbox by pressing 'C' (capital c, since lower case c is the 'change directory' command). Mutt will ask you "copy message to folder?" Type the address of where you would like to save the file, for example ~/Desktop/New, or equivalent.[3] That file will include the entire set of internet headers that traveled with the message, which is probably more than you wanted. In that case, press esc-C (escape and then capital c) to access the "save decoded copy" function, which does the same thing but in a more reasonable format. Both of these functions leave the message in your mailbox.

If you want to simultaneously remove the message from your inbox, use esc-s (escape, followed by a lower case 's') for "decode save," which will remove the message from your inbox once it's created the text file for you. I find this very useful for archiving individual messages.

[3]Note that the variable 'mbox_type' determines whether a text file will be created in that location or an annoying maildir style folder hierarchy. If you get folders, type ":set mbox_type=mbox" and then try again.

2.8 Operating on Multiple Messages

2.8.1 Selection Patterns

This is where mutt starts getting powerful: lots of operations can affect many messages at once. You can tag a group of messages (see section 2.8.2 below) and do something to them (delete them, move them, print them), or you can limit the number of messages shown in the index view (see section 2.8.3 below) by providing criteria for what you'd like to show. Both of these require an understanding of selection patterns. A brief description of patterns follows, and a complete list can be found in section A at the end of this document. If they're confusing, don't use them, but if you can get the hang of using patterns you can do lots of interesting things that will speed up your emailing. Two of these tricks are group actions on a set of email (you tag them and then do what you want with them), and limiting the messages shown to you in the index based on some sort of criteria. These are both described in the next two sections. But first, here are a couple of simple ones.

Limiter	Limits messages shown to:
~t	mail sent TO
~f	mail sent FROM
~c	mail cc:ed TO
~U	unread messages
~R	read messages
~N	new messages
~O	old messages
~b	messages that contain a certain expression in the body
~B	messages that contain a certain expression in any header or the body

Figure 7: Simple Patterns

You can use multiple criteria by separating them with a space, like this. For example, the following selects all messages that were sent by someone with Wood in the address field AND that were sent within the last two weeks:

```
~f Wood ~d<2w
```

and the following would select any message that is both new and from Ericka:

```
~N ~f Ericka
```

The logical "OR" is indicated with the pipe symbol, so the following would select anything with the word "viagra" or 'OEM" in the subject line (e.g., for deletion):

```
l viagra|OEM
```

and the following would select all messages from Wood or Feldman or Dordyke:

```
~f Wood|~f Feldman|~f Dordyke
```

Watch out for spaces after the pipe symbol, which will interfere with your expression.

2.8.2 Tagging and Flags

To operate on many messages at a time you need to tag them first, then choose your command. Use the 't' key to tag messages. It's a toggle, so hit 't' again to untag them. Once you've selected the messages you'd like to operate on, preface your command with a semicolon to indicate the command should be applied to all tagged messages. For example, the command ';s=Nicaragua' will save all tagged messages to the mail folder by the name of 'Nicaragua' and ';d' will delete them.

You can tag messages individually if you like, but it's faster to use a pattern to select them. Shift-T tags messages that match a pattern, and control-T untags messages that match a pattern. Escape-T will tag a whole thread, if you've chosen to display your messages based on threads. Once you've tagged a group of messages, you can do anything you want to them. For example, you can even tag several messages and hit 'r' for reply and mutt will combine the text of all those messages into a single message and, once you've composed your message text, send it to the authors of each of those messages. This is useful if you correspond with a person who sends you 6 separate emails about 6 separate topics.

2.8.3 Limiting Messages Shown

Limiting is another way that patterns increase your efficiency. More often than not, you've got too much mail in your box to deal with. Limiting how much of it you see is convenient and helpful, and only a keystroke away. The 'l' command (lower case L) limits the messages shown in the index. It doesn't delete anything, it just limits which messages are presented in the index according to your criteria, which you can specify using regular expressions or certain keys.

For example, hit 'l' and then type the word 'proje' and hit return. Of all the messages in the index, only those that have the expression 'proje' somewhere in the subject or address fields will be shown. To remove the limitation, hit 'l.' (that's L followed by a period), which shows messages that contain any character at all, that is, every message. See section 7 for a complete list of patterns. For example, to limit by date, l ~d>2w to limit the index to all messages whose received date (~d) is greater than two weeks (>2w), or set l ~N to see only messages that have been flagged 'new.'

If you forget what your criteria for limiting was, or would just like to confirm whether or not you are limiting them, pressing Esc-L will reveal in the status bar what the current limiter is.

The ~d sequence limits messages based on date criteria. This is highly useful. An example is worth a thousand words, so let the following serve as guides:

Limiter	Limits messages to:
l ~d 20/1/05-31/10	all messages whose date is between 20 Jan 1995 and 31 Oct of the current year
l ~d<1d	all messages less than one day old, that is, all messages received today
l ~d 15/1/2001*2w	all messages plus or minus two weeks from 15 Jan 2001

Figure 8: Criteria for Working with Dates

2.8.4 Using Patterns to Archive Groups of Messages

By tagging groups of messages and saving them to off-line folders you can easily archive in a format that remains readable rather than some mysterious binary format. The format of that file will either be a mailbox

(i.e., a single, flat file), a maildir like those on IMAP systems, depending on the default for your system (see sec. 3.7 on how to set this variable). I like to use tags to gather messages by date range and archive them to a flat file, then zip the file to save space on my hard drive. When I want to read through archives I can do so by starting mutt from the -f command.

First tag the messages, using `T ~d>30d`, which will tag all messages greater than 30 days old. THen save them to a file called sept-2006 using the command `;s=~/Documents/Mailarchives/sept-2006`

To read the messages in that archive, treat it like a mailbox and read it using `mutt -f ~/Documents/Mailarchives/ sept-2006`.

2.9 Using the Address Book

Mutt allows you to determine who you'd like to send the message to either before or after launching your text editor to compose the message. You are free to type in the entire email address, of course, but that isn't very effective. To access the addresses you have stored in your mutt address book, simply hit the tab key at any point where an address is requested of you. If you type a couple of letters before hitting tab, mutt will limit the entries in the address book to those that match ('eri' will match Ericka, Erico, Erik, Erin, and so on). Navigate to the address you want or hit the forward slash key to search for one, and press Return to have that address be added to the list. You can select multiple addresses using the space bar.

While looking at the message index you can view the sender's email address by pressing the @ key. On the status line at the bottom of the screen you should see the sender's complete email address. Hit 'a' to add that sender to your address book. You will be asked to confirm the name and email address of the sender (which is a great time to modify it for your own purposes if you'd like to do so).

In the context of more complicated programs that integrate the address book with other software (like instant messaging clients: Kontact/Kmail/Kopete on Linux and Addressbook/Ichat/Mail on Apple Mac OS X), the rudimentary mutt address book is somewhat of a disappointment. It is simply a text file containing a list of names and email addresses organized by aliases (nicknames) that you assign them. Each entry, one per line looks like this:

```
alias randall_wood Randall Wood <rwood@therandymon.com>
alias dogcito Dogcito and Oso <dogcito@zafiro.com>
```

You can add additional entries as described above, edit the text file directly any way you like, or try integrating an add-on address book with a bit more functionality. In section 3.5 I'll discuss how to configure the mutt address book, and in 3.5.2 below I'll discuss how to interface with more powerful address book components.

3 Configuring Mutt

3.1 Basic Configuration

Mutt is a mail user agent (MUA), and makes no effort to provide native ability to download, upload, or compose messages, relinquishing responsibility for those tasks to programs better suited to the job. Neither

does it deal with HTML mail, attachments, or help you click on a URL in the text to be taken to the relevant web page. Rather, mutt interfaces with lots of other programs that already perform these tasks admirably. This means to use mutt you have to configure several different programs, including sendmail or equivalent for uploading (sending), fetchmail for downloading (assuming you have a POP3 account rather than IMAP), the lynx text browser for viewing HTML messages, and urlview for dealing with URLs in mail. In general, you will need the following configuration files on your system:

.muttrc Defines all settings for Mutt itself

.mailcap Tells mutt which software to associate with different attachments

.fetchmailrc For downloading mail from a POP3 server, if that's how you are connecting

a mail spool file probably /var/mail/spool/$USERNAME

an alias file of your choosing, such as .mutt-alias. This is your address book

The spool file is where mutt expects to find mail that's been downloaded from your ISP (i.e. your inbox). You'll also want to choose and configure your favorite text editor. Unix and Linux systems come with several text editors already installed, so it's just a matter of your choosing which one you like. Mutt also requires an address book called an alias file, and will help you develop one as you go through your mail, if you like. More on each of these things below.

Your .muttrc file is a configuration file that should be stored in your home directory. If you don't have one in your home directory mutt will utilize the system-wide configuration file at /etc/Muttrc (note the capital M). You can copy that file to your home directory, calling it .muttrc (note the period; it's a dot-file). This is the file you will modify to set up mutt to your liking. Alternatively, you can start a new .muttrc file using the text editor of your choice using the code in this document as a base.

The .muttrc file contains code that determines mutt's keybindings (which keys perform which function), code that determines what mutt looks like, including colors, and code that determines how mutt behaves. First of all, the hash mark (#) is interpreted as a comment sign, so any text after a hash mark is ignored when mutt processes its files. If you want to try a test .muttrc you can name it something different and start mutt from the command line with `mutt -F TESTFILE` to use it in place of your .muttrc. However, you can also test out commands one by one once mutt is running, by pressing the colon sign and entering the command. You wouldn't want to do that for large sets of commands, obviously, but if you simply want to try something both ways, you can do so in this way. Still another way is to get mutt started, then make edits to your .muttrc file, and have mutt reload the configuration file by entering `:source .muttrc`. This is my favorite way of testing configurations as I go.

3.2 Setting Up Your Mail Accounts

Unless stated otherwise, all these configuration statements go in your .muttrc configuration file. No matter what your email set-up, you will need to set the variables `mailboxes`, `spoolfile`, and `folders` to tell Mutt where to find your mail account, inbox, and folders. You will probably also need to set `record` to choose where to store copies of your sent messages.

3.2.1 Downloading from a server (POP3)

Getting your mail: In this scenario, your ISP has provided you with a mailbox, and you download its contents and deal with your mail offline. If this is your situation, mutt expects to find your mail waiting for you on your system at the location /var/spool/mail/USERNAME, and you will use Fetchmail to retrieve it from the server.

First, Fetchmail: Fetchmail is included on all Linux distributions worth their salt these days. In your home directory you should create a dot file (configuration file) by the name of .fetchmailrc. In it, provide your download information. Fetchmail is well documented, so either start with your system's sample fetchmail file (/etc/fetchmailrc) or read the man page for fetchmail to get an idea of all the things fetchmail is capable of. For any POP3 account I've ever had, all I've needed has been something like the following. Replace the UPPERCASE terms with your own information; USERNAME is the name by which your ISP knows you; LOCALUSERNAME is the name by which you log onto your Linux box.

```
poll MAIL.MYISP.COM with protocol pop3 user USERNAME
      with pass PASSWORD is LOCALUSERNAME here;
```

To get your mail, open up a terminal and download it by entering the command `fetchmail`. Some distros like SUSE set up your machine so your mail is automatically downloaded every time you establish an internet connection. When you sit down at your computer you can open up a terminal and first run fetchmail, then run mutt. Or you can start with mutt, and run fetchmail by entering `! fetchmail`. The exclamation point is the mechanism for getting to a shell from within mutt. After `!` you could easily input any shell command you like (less, sort, ls, vim, etc.)

Now, mutt: If your log-in name on your system is randymon, mutt expects to find a text file with appropriate permissions (that is, user can read and write) at /var/spool/mail/randymon. If that file doesn't exist, root will have to create it for you. Remember, it's a file, not a directory, so you can create it with the "touch" command and then set the permissions using "chmod."

```
mailboxes  /var/spool/mail/randymon
set spoolfile=/var/spool/mail/randymon
set folder="$HOME/Mail"
set record="$HOME/Mail/Sent"
```

Sending your mail: If you're working offline, you will probably be using Sendmail or Postfix as a mail transfer agent (MTA), or something equivalent, and your Linux box will send off the queued mail whenever you connect. See section 3.3 for other options.

3.2.2 Accessing your Mail at the Server (IMAP)

Getting your mail: If you use IMAP, your mail never gets downloaded. There are lots of advantages to this method. Use something like the following in your .muttrc

```
mailboxes imap://myfirstisp.com
set folder="imap://myfirstisp.com/INBOX"
set imap_user="myusername" #your IMAP user name or login
set imap_pass="password" #your IMAP password
```

If you are uncomfortable about having your IMAP password stored in plain text, set the permissions for .muttrc using the "chmod" command so only you can read and write to the file, no one else. In this same section of your .muttrc, show mutt where to find your IMAP inbox. This is sometimes not apparent, even when you've specified where to find your mail.

```
set spoolfile="imap://myfirstisp.com"
```

Header Caching: Mutt suffers from one of IMAP's deficiencies: every time you enter a new IMAP folder Mutt has to reread the contents of the entire folder in order to present you with the index. For folders with a lot of messages, or any folder if you're working over a slow connection, that can be the source of a very lengthy pause. Caching the headers takes care of the problem, by providing mutt with a mechanism for storing the headers locally; mutt then only needs to compare the contents of your IMAP folder with the local cache, and react accordingly. For most IMAP users, this is a huge improvement. Prior to about 2008, header caching was a patch you had to compile in yourself. These days, it's standard (it came automatically with my SUSE 9.2 distro). The header cache can take two forms: a folder of files, or a single file. If .mutt-headercache (or whatever you call it) is a file, your cache will be a single file. I prefer it to be a folder of files, so simply `mkdir .mutt-headercache` to do so. Make it a dot file so you don't have to look at it every day in my home folder. Once you've created a home for the cache, simply tell mutt where to find it, as follows:

```
set header_cache=/home/randymon/.mutt-headercache
```

Sending your mail: Again, you'll either be using Sendmail or an equivalent, to send email. But here, Mutt's internal SMTP capability is also available to you, since you will have a live Internet connection while you're working. The trick in this scenariois to ensure you've configured your Sent Mail folder correctly so that all your sent messages are registered on the server instead of simply being saved locally. See section 3.3.

3.2.3 Offline IMAP

Getting your mail: From a user's point of view, Mac OSX's Mail program got it right: your IMAP account is cached locally, and updates changes whenever you connect. It does away with "I'm not connected to the 'Net so I can't access my old mail" and basically allows the user not to care what kind of mail account he's using. Mutt users can now do the same. Offlineimap is a package of Python scripts that replicates your entire IMAP account locally, sub-folders and all. The first time you sync, it can take some time, but afterwards the program runs quickly, and you can even set it to run in the background.

First, download and install offlineimap. It requires a somewhat modern version of Python. Next, give offlineimap a place to store its local cache (this will turn into a gigabyte- or greater-sized directory, so choose

carefully): `mkdir .mailcache`. The man page will walk you through the next steps, but essentially, you need only to declare the parameters for the local and IMAP accounts, and then execute the program periodically. Here's a sample offlineimap configuration:

```
[general]
accounts = personalmail

[Account personalmail]
localrepository = local
remoterepository = remote

[Repository local]
type = Maildir
localfolders = ~/.mailcache
sep = /
[Repository remote]
type = IMAP
remotehost = MYISP.WHATEVER.NET
remoteuser = MYLOGINNAME
ssl = yes
remotepass = MYPASSWORD
```

Finally, since you will now be accessing not your IMAP account but the local, replicated repository, your .muttrc should point to it as follows:

```
set folder=$HOME/.mailcache/INBOX/
set spoolfile=$HOME/.mailcache/INBOX
set record="=Sent"   # or whatever your sent mailbox is called
```

Sending your mail: You have two options: one is to use the sendmail installed on your system, having configured it for offline mode (mail is queued locally, and sent only when your machine detects a network connection). Your second option is to use putmail.py-queue, a partner script to putmail. Putmail.py-queue stores outgoing mail in a folder, and then sends it to putmail one by one when you instruct it to dequeue. In that case, you would use `set sendmail="/usr/local/bin/putmail_enqueue.py"` in your .muttrc.

When you get a 'net connection, run `putmail_dequeue.py` to send out all the stored mail you've accumulated.

3.3 Sending Mail: Sendmail and Friends

Sendmail: Before version 1.5.17, you were required to configure a mail transport agent, like sendmail. For old time's sake, and for those using older versions of mutt, this is how to do so. Provided sendmail, postfix, qmail, or another SMTP mail sending program has been installed and configured on your system, simply tell

mutt where to find the sendmail program. In most cases, sendmail is located at /usr/sbin/sendmail, so the following line should suffice. Postfix even provides a simlink at that location to deal with mail programs expecting to find sendmail there. So once you know where to find sendmail, put a line in your .muttrc file that tells mutt where to hand off written mail for sending, as follows: `set sendmail=/usr/sbin/sendmail`

However, installing and configuring Postfix or Sendmail is not fun in an age when any self-respecting ISP requires authentication to stem the spamflood, so choosing a distribution that does this for you, like SUSE, will take a lot of pain out the process (Ubuntu users, you are out of luck, change distros or learn how to configure sendmail). If you're just a casual emailer instead of a systems administrator, and just need a simple and easy way to get your mail to your ISP, perhaps a full-blown Postfix configuration is overkill. There are several alternatives created for the likes of mutt users who just need an easy way to send mail to their ISP, and nothing fancier than that.

Mutt's own SMTP: Starting with version 1.5.17, mutt now – miracle of miracles – has the capability of sending your messages without the need for another program like sendmail. This goes essentially against the mutt philosophy, but was requested by enough users that the decision was made to include SMTP capability. To take advantage of this, in your muttrc file you would add your log in information in the following format:

```
set smtp_url="smtp[s]://[user[:pass]@]host[:port]/"
```

Let's say you have a mail account whose SMTP address is outmail.mailhost.com, where you log in using the name 'squeaker' with password 'bear.'

```
set smtp_url="smtp://squeaker:bear@outmail.mailhost.com"
```

SSMTP: The first is "ssmtp" ("simple SMTP"),a package that allows you to send mail via the SMTP protocol to your ISP, authenticating with a username and password. For most people, that's all you need to do anyway, and choosing this option simplifies the process of installing mail transport agents that were really intended to go on servers and handle immense volumes of mail, spam, and bounces every day.

If that's the mechanism you choose, your .muttrc file should look like this:

```
set sendmail="/usr/sbin/ssmtp -audUserName@domain -apSecretPassword"
```

Putmail.py: Just as easy, if not easier, is Putmail.py, a tiny python script that provides a dead simple interface for sending mail. It's a simple 13 KB download and an even simpler 5 line configuration file, and you are up and running. Once it is working you can even configure other software to use it, such as the slrn newsreader. This is now my preferred configuration on the Macintosh, which lacks easy postfix configuration utilities, while on SUSE I stick with YaST2 and Postfix.[4] There are two packages: putmail.py and putmail-queue.py. The latter allows you to build up a queue of outgoing mail while you're offline, then send it all at once. A putmail.py configuration file is as simple as the following. In most cases, your email address and username should be the same:

[4]Thanks to Kevin Neely for convincing me to take another look at Putmail.py. It so impressed me I'm taking a much harder look at Python in general!

```
[config]
server = OUTGOING.MAIL.SERVER
email = YOUR@FROMMAIL.ADDRESS
password = PASS
username = USERNAME@FROMMAIL.ADDRESS
```

Others: There are several other light SMTP agents that do what the ssmtp program does, including esmtp, msmtp, nullmailer, and others. See the Mutt Wiki for a complete list (see section 5). Otherwise, welcome to the wonderful world of configuring mail transfer agents. That task is beyond the scope of this guide. The most useful resource for configuring postfix for the purposes of a mutt user (that is, you just need it to send mail to your ISP) is entitled "Postfix State of Mind." See the Appendices for its URL.

3.4 Your Sent Mail box

This depends a lot on whether you will be mostly working online (IMAP) or offline. If you are working offline (downloading via POP3 or equivalent), the following is good enough:

```
set copy=yes #keep copies of outgoing mail.
set record="/home/randymon/Mail/sent-mail "
#This is for locally stored mail
```

If you're using IMAP, then you should store your sent messages on the server, as follows:

```
set record="=Sent Items"
#This is for the 'Sent Items' folder on an IMAP account.
# Note the equals sign, which makes it clear
# this a subdirectory of the inbox.
```

And if you're using offlineimap, you can store your sent messages in the local Sent Mail repository, and will sync up to the IMAP server when offlineimap runs. Use

```
set copy=yes #keep copies of outgoing mail.
set record="/home/randymon/.mailcache/INBOX/Sent"
```

Lastly, an advanced trick: you can specify which folder to use on a per-message basis using send-hooks (see section 4.10).

3.5 Configuring Your Address Book

3.5.1 Alias File

This is the simplest configuration, and the default. Out of the box, Mutt stores your email addresses in any file of your choosing in the following format:

```
alias linux_lover Linux Lover  <linuxlover@yahoo.com>
```

where linux_lover is an alias you choose for the address, the next couple of words are the person's real name, and everything between brackets is the email address. We "source" that file (identify it as a file to be read into memory when using mutt to make it available by identifying it as our alias file (again, if it doesn't exist, you'll have to create it yourself first).

```
set alias_file="~/.muttalias"
```

And here's a neat trick: we set `reverse_alias` so that any mail sent to us by a person in our address book is shown using the name in our address book, not the name their mail goes out with. That is, if our correspondent is really an idiot, our alias can read

```
alias Idiot  Firstname Lastname   <coworker@myworkplace.com>
```

and our mailbox will show mail from that person as coming from the name "Idiot". A better use of this trick is to deal with people whose names aren't easy understandable, like x36gd9o_33a@hotmail.com. And no, the other person will not know you've labeled him 'Idiot,' it simply changes the display of his name in your mailbox, not in your messages.

```
set reverse_alias #use names from my own address book, if they exist
```

There are more sophisticated ways to manage your address book, by using external programs like abook. See section 3.5.2 for details. If you'd like more functionality in your address book without leaving the console, think about helping mutt interface with something more powerful. There are several small console-based address books with which mutt is able to interface, but unfortunately no real way to interface with something truly powerful like Apple's Addressbook or Kmail's KAddressbook.

3.5.2 Abook

Abook is a simple console address book that stores all your information in text files and interfaces well with mutt. Written by Jaakko Heinonen, it is undergoing active development as of September 2006. Abook is easy to configure and use. It does not, however, understand the vcard standard file format: this is its most serious limitation. Also, in earlier versions, abook ran parallel to your alias file and you had to manually syncronize them every so often, which was not very convenient; fortunately, that has been dealt with, and these days, for a decent address book and an easy interface with mutt, abook is hard to beat.

To use abook, first download and install it (see section 5.2 for the link), and get familiar with how it works by running it from a console. It's pretty straightforward, and you can either type in a couple of addresses to practice with or import your mutt alias file. Now that you've got some addresses to work with, integrate it with mutt.

The first step is to remove the links to the mutt alias file, so assuming you called your alias file .mutt-alias, make sure to remove the following two lines from your .muttrc:

```
set alias_file=".mutt-alias"
source .mutt-alias
```

And now add the following two lines in their place:

```
set query_command= "abook --mutt-query '%s'"
macro index,pager A "<pipe-message>abook --add-email-quiet<return>"
            "add the sender address to abook"
```

The first line sets up the link to abook for extracting addresses from your addressbook, and the second one creates a macro so that when you press 'A' just about anywhere, the email address of that message's sender will be sent to abook.

Lastly, get used to using "query" functionality instead of alias functionality, as mutt treats them differently. Let's say you start a new message by pressing 'm' in the index view. Mutt presents you with "To: " and waits for you to enter the address of your recipient. Normally you'd press tab at this point for a quick search of the aliases in your alias file. Instead, now you press Control-T in order to do the same thing, but via a query to abook instead of your alias file. Put the cursor over the person you'd like to write and press enter to start the message. If you'd like to several people from the abook, tag them using the 't' key and then press ;m to have them all placed on the "To:" line of your message. You can add additional queries to the same line by pressing 'A.'

For example, type Control-t to start a query, "Do" to search for names that start with "Do." You select a few by pressing 't' and then press 'A' to search for additional names using a different criteria. Press "Ri" to search for names that start with "Ri" and select them in the same way. When you've tagged everyone you'd like to mail, press ";m" to send all those addresses to the "To: " line. Repeat this process for the "Cc: " line and so on.

Some people like to use the abook and the alias file in parallel, keeping good/important addresses in the abook and using the alias file to collect addresses from mail they read. This is a bit inelegant since it forces you to manually synchronize from time to time, but has its advantages, since, for one, the alias file allows you to include a mailing list composed of other names present in the alias file, while abook does not. Nothing is simple, but take a look at `http://wolfermann.org/abook-autoexport` for one way of doing so.

3.5.3 Querying an LDAP Server

Mutt's query command (Q, pressed at any prompt, or control-T after you've input a couple of characters to search for) is powerful and impressive. To query an LDAP server, first download the mutt-ldap.pl script (find a link to it at the mutt wiki) and save it to someplace where you can execute files (I saved it to ~/bin). You need to update just four lines in the perl script, and you are good to go. Increasingly, this is how I deal with addresses.[5]

[5]Of course, since it's just an executable perl script, you can call it from the command line at any time. From the command prompt, just enter `mutt-ldap.pl hoozit` to search your LDAP server for a user named hoozit.

3.5.4 The Little Brother Database

You can get the Little Brother Database (lbdb) from its sourceforge site (see 5). Set mutt to query the LBDB for addresses by adding the following to your muttrc:

```
set query_command="lbdb' '%s'"
```

The LBDB can query different sources of information to manage your addresses. See the Mark's Mutt Fan and Tip Page (Resources) for more information on how to configure the LBDB with mutt. In particular, the LBDB is a useful way for Mac users to get Mutt to interface with the Apple Addressbook that comes standard on Mac OS X. Vincent Danen does a tremendously good job at describing not just how to get this link to happen but also how to set up and use Mutt on Mac OS X in general. His website is listed in the Resources section of this guide.

3.5.5 Rolo Addressbook

The Rolo Address book purports to do what Abook does, but using the vcard standard. I haven't used it so I can't say, but working with vcards is an important step toward being able to import and export addressbooks from other software packages. See the references in section 5 for the source of this software.

3.6 Choosing an Editor

Mutt doesn't come with its own facility for writing email because on Unix/Linux systems (etc.) more good quality text editors are available than you could ever hope for. So it's just a matter of choosing whichever one you like the best. Then, whenever you go to write a new email, mutt will launch your editor for you. When you exit the editor, that text will be transfered to mutt and incorporated into your email ready to send. Which editor you choose is a personal decision. Set it with the command "set editor=" and enjoy. The following lines are some samples; note four of them have been commented out leaving emacs -nw as my editor of choice. When I get sick of emacs, I just comment out that line and uncomment another one, and I'm ready to go. I tend to mostly switch between emacs and jed, as jed is a little faster and lighter-weight, but emacs is more feature rich. Again, it's personal.

A final note: the editors I use tend to be console based, so no external windows are launched and I can work as easily at a virtual console as I can in an xterm window. If you don't use the console and live in the XWindow environment, you can launch a GUI editor like emacs, xemacs, nedit, kate, kwrite, gedit, and so on just as simply, paying only the price of a slightly slower launch time. To set your editor, choose one of the lines below and remove the hash mark at the beginning, leaving the other lines as comments:

```
# set editor="vim +':set textwidth=77' +':set wrap'"
#use vim with wordwrap at 77
  set editor="emacs -nw"
# set editor="joe"
# set editor="jed"
# set editor="pico"
```

3.7 General Settings

In this section of the .muttrc file we establish a couple of general things. “set charset” determines which character set your terminal is using. In many cases it’s not important, but if your display is showing funny characters in place of accented characters like á or ñ or even the little arrows that show where the messages have been threaded, this is how you can deal with it.

”unset beep” disables annoying beeps. “confirmappend” is a setting that has mutt ask you every time you save messages to a given folder, and “use domain” is an option that adds the local domain name to any email sent to an address without an @ sign (so if you send a message to “jeremy”, not “jeremy@sample.com” mutt will change the name to “jeremy@localhost”). That’s not usually useful, so I usually unset it. Turning off the recall function prevents mutt from asking you at the start of every session if you’d like to resume a postponed message (provided postponed messages exist, of course). Since I infrequently start messages I don’t finish, I turn off that attribute. And you can always resume postponed messages any time you want by pressing Control-r in mutt. Finally, ‘push <show-version>’ simply tells mutt to display its version number upon start up, just for fun.

```
# Starting up, General Settings
set charset=UTF8
#depending on our environment it could be ISO-8859-1 (Western Latin)

set mailbox_type=maildir
#default mailbox format (could be maildir, mbox or several others)

unset beep
#don’t need no beeping software

set print_command="lpr -p"
#Send your message to the line printer. If you’ve added the mutt-print
#package, use the following line instead:
set print_command="mutt-print %s"

unset confirmappend
#append what?

unset use_domain
set recall=no
#don’t resume postponed messages

push <show-version>
#get Mutt to show its version on startup

set quit=ask-yes
#Ask before quitting, but default to yes

set nomove
#keep messages in MAIL, don’t ask about moving them anywhere when quitting
```

```
unset mark_old
#don't mark messages as old when I pass over them with the cursor
```

These are just several of the hundreds of configuration parameters over which mutt gives you control. This is what people mean when they say mutt is highly configurable. For more settings and a glimpse of what your options are, read the muttrc man page or mutt's online documentation (see sec. 5 for the URL).

Now let's look at the settings that affect our general interaction with mutt. The following lines of code establish my name and email address. 'set include' gives you the opportunity of automatically stripping out the message you're replying to in the text of your email. Mutt can also by default hide the "Cc" and "Bcc" lines of the address if you don't typically use them. I don't BCC anybody, so in my .muttrc I have unset that option, leaving only "cc"s. "unset self" is a great feature I wish other email clients would implement. When replying to a group email, mutt is smart enough to recognize and remove your address from the list of recipients so you don't accidentally send yourself the message.

The following two lines deal with attribution of replies and forwarded messages. You could change this so that in an email reply, underneath your text it reads "On July 4, 2005, you vomited up the following nonsense:" or equivalent.

```
# Composing Messages
set from="Randall Wood <my_email@address.com>"
#My name and email address as shown in other people's inboxes
set include=yes
#include message in replies
set askcc
#include a cc: line on which to add recipients
# set askbcc
#include a bcc: line
set fcc_attach
#forward attachments
unset reply_self
#don't include myself when replying to a group
set attribution="On %d, %n wrote:"
set forward_format="Fwd: %s"
set indent_str="> "
#indented text prefaced by this string.
set postpone=ask-no
#default for postponing a message is to confirm, default to no
set tilde
```

3.8 Dealing with HTML Messages

Let's start with HTML. Mutt alone doesn't handle HTML text on its own, but since the console web browser lynx does, we can leverage its power to filter out the HTML markup and present it to us in text format. This is extremely useful. To do this, as well as deal with other attachments, we need to create a .mailcap file in our

home directory that lists possible MIME types (attachment types) and how to deal with them. Your .mailcap file is a simple text file and may already exist in your home directory, depending on your distribution. If it does exist, simply add the following line:

```
text/html; lynx -dump -force_html  %s;  needsterminal; copiousoutput
```

Once your .mailcap tells mutt how to process HTML files, you must tell mutt to first source the .mailcap, and second to process the HTML and present it inline, instead of treating it as an attachment to be shown separately. This takes away the HTML mystery of your friends who insist on using hotmail: mutt will filter out the text and simply present it without the formatting. If you respond to or forward the message, the text will remain and the formatting will be permanently removed. People who tend to gravitate towards console email clients tend to like this ability to strip out all extraneous formatting that makes email large and unwieldy. Once you've modified the .mailcap file, add the following lines to your .muttrc:

```
set implicit_autoview
auto_view text/html  application/x-pgp-message
set mailcap_path="~/.mailcap"
```

The first line tells mutt to automatically show everything it knows how to show without being prompted, the second line tells mutt which types of files to show as part of the body text, and the third line tells mutt where to find the mailcap.

3.9 Dealing with Attachments

The mailcap is the place to indicate to mutt (and other programs, incidentally; such is Unix) how to interface with other programs to process attachments. Images are an important one to get right. Add something like the following in order to process PDFs and graphics images.:

```
application/pdf; acroread %s
image/jpeg; xv %s
image/gif; xv %s &
image/GIF; xv %s &
image/JPG; xv %s &
image/jpg; xv %s &
```

Above, I'm using the program 'xv' to deal with graphics, because it's super fast relative to the KDE and Gnome equivalents and SUSE provides it. But it's not included on a lot of distributions (including anything based on Debian) because of licensing concerns, so you can substitute kview, gwenview, eog (Eye of Gnome), or something else for xv.

3.10 Dealing with Word Doc Attachments

The most important one to include is also the one that can make mutt so much more useful than other mail clients: an interface with antiword. This is a program that reads Microsoft Word documents and translates

them to plain text. It's highly useful on its own as a document tool, but combined with mutt it becomes more powerful still. Assuming you have installed antiword on your machine, add the following to your mailcap file:

```
application/msword; antiword %s ; copiousoutput
```

And Word doc text will be displayed inline. This is an extraordinarily useful trick! If your distribution doesn't include antiword, try the application `wvText` instead, which works just as well.

This doesn't solve all your problems, but should solve many of them. Some broken webmail email applications don't label MIME types appropriately, which will hinder Mutt's ability to appropriately determine which application to use to open the file.

3.11 Dealing with URLs in Messages

Many messages have URLs embedded in the text. Urlview is the right tool to deal with them. You can configure it to launch a graphical browser like Firefox, Opera, or Konqueror if you have the X Window system running and launch lynx or links if you don't. Fortunately it's easy for urlview to distinguish whether or not you're using X and proceed smoothly from there.

First, install urlview. Second, add the following lines to your .muttrc:

```
macro index \cb |urlview\n 'call urlview to extract URLs out of a message'
```

The macro above sets the key binding control-B to launch urlview (more about macros in section 4.4). Try it once and you'll see how nice a system it is.

4 Advanced Customization

4.1 Using Mutt with different configurations

If you wind up using mutt sometimes with IMAP and sometimes with offlineimap, or once for your home account and once for your work account, you will need different configurations. My simple solution is simply to use different .muttrc files, and create aliases to run them. For example, create .mutt-home, .mutt-school, and .mutt-home-offline files. Then in your .bashrc file, create aliases, so that `homemutt` becomes a command that runs `mutt -F .mutt-home`. In your .bashrc, that would be `alias homemutt='mutt -F .mutt-home'`. Do the same for your other configurations.

4.2 Configuring Colors

Colors are available to you in terminals that support them, including most Linux virtual terminals, and many terminals for the X Window system like xterm, aterm, and so on. You may want different color schemes for different circumstances though: xterms are black-on-white unless you specify otherwise, but virtual

consoles are white on black. In addition, aterm, gnome-terminal, and konsole are all terminal emulators capable of using transparency and background images, both of which are very pleasant effects. So instead of committing yourself to colors in your .muttrc file it's better to put groups of colors in additional files and source them as necessary. For example, create .mutt-clear, which will be the color configuration when you use mutt in a transparent terminal, .mutt-white for black on white scenarios, and .mutt-black for white on black scenarios. Then once mutt is running you can source the one you would like to use by entering:

```
: source ~/.mutt-clear
```

Note the colon at the beginning of the line, which indicates to mutt you'd like to enter a command. Alternately, you can simply source the file in your .muttrc file by adding that line to your .muttrc. Color works as follows: You indicate which object you'd like to define a color for and its foreground and background colors, plus a regular expression if the color is to be conditional (for example, to provide color for text that looks like an email address). You can remove a color with the uncolor command, something I've never had ocassion to do. The following three lines are examples of the format, and you can then choose an object from chart 9 and a color from the following list: white, black, green, magenta, blue, cyan, yellow, red, default, colorx. The foreground color can also be prefixed with the keyword "bright" to make the foreground color boldfaced (e.g., brightred). If you run mutt in a clear terminal, make sure to choose 'default' as your background color in order to preserve the transparency.

```
color object foreground background [ regexp ]
color index foreground background pattern
uncolor index pattern [ pattern ... ]
```

As usual, an example is worth its weight in gold. The following are my color selections, including some that I copied from the defaults that come along with the Debian package for mutt. In general, this provides a black background, white text, attachment labels in yellow, quoted text in green, and deleted messages in red. In addition, email addresses and URLs are highlighted in bright colors in the message body.

```
color normal white black
color attachment brightyellow black
color hdrdefault cyan black
color indicator black cyan
color markers brightred black
color quoted green black
color signature cyan black
color status brightgreen blue
color tilde blue black
color tree red black

color index red black ~D
color index magenta black ~T

#color header brightgreen black ^From:
#color header brightcyan black ^To:
```

Name	Description
attachment	any attachments present
body	match regexp in the body of messages
bold	highlighting bold patterns in the body of messages
error	error messages printed by Mutt
header	match regexp in the message header
hdrdefault	default color of the message header in the pager
index	match pattern in the message index
indicator	arrow or bar used to indicate the current item in a menu
markers	the "+" markers at the beginning of wrapped lines in the pager
message	informational messages
normal	normal (not quoted) text
quoted	text matching $quote_regexp in the body of a message
quoted1 ... quotedN	higher levels of quoting
search	highlighting of words in the pager
signature	any text beneath the signature line
status	mode lines used to display info about the mailbox or message
tilde	the "~" used to pad blank lines in the pager
tree	thread tree drawn in the message index and attachment menu
underline	hiliting underlined patterns in the body of messages

Figure 9: Parts (Objects) of Messages that can be Highlighted

```
#color header brightcyan black ^Reply-To:
#color header brightcyan black ^Cc:
#color header brightblue black ^Subject:

color body brightred black [\-\.+_a-zA-Z0-9]+@[\-\.a-zA-Z0-9]+
# identifies email addresses

color body brightblue black (https?|ftp)://[\-\.,/%~_:?&=\#a-zA-Z0-9]+
# identifies URLs
```

Here's one way to make it useful. Let's say you receive mail at both `penguin@iceberg.net` and `penguin@school.edu`, and the latter account simply forwards all received mail to your iceberg.net account. You can different between the two types of mail in your inbox by including in your .muttrc `color index green default ' C penguin'`. Any mail where the word "penguin" appears in the to: or CC: lines will be shown as green.

4.3 Rebinding Keyboard Shortcuts

By binding keys to certain functions you can add shortcuts to maneuvering through your message list. For example, 'j' and 'k' are configured out of the box as up and down keys for moving around in the index. But you can just as easily bind them to something else if you like (for example if you use a non-standard

keyboard layout like Dvorak). The ‘bind’ command allows you to configure mutt the way you like to work. The command takes the following form: “bind ‘map’ ‘key’ ‘function.’ ”

Maps can be one of the following: generic, alias, attach, browser, editor, index, compose, pager, pgp, and postpone, and determines in which context the key is to be bound, i.e. when you are maneuvering around the index, when you are in the pager, when you are searching for an alias in your address book, and so on.

Key is the key (or key sequence) you wish to bind. To specify a control character, use the sequence \Cx, where x is the letter of the control character (for example, to specify control-A use “\Ca”). Note that the case of x as well as \C is ignored, so that \CA, \Ca, \cA and \ca are all equivalent. Mutt allows you to use non alpha-numeric keys on a lot of platforms, but I was unable to get the F keys (F1, F2, etc.) to work on my Macintosh, so it’s not universal. Using the brackets shown below you can access any key listed in table 10 below:

Code	Produces:
\ t	tab
<tab>	tab
\ r	carriage return
\ n	newline
\ e	escape
<esc>	escape
<up>	up arrow
<down>	down arrow
<left>	left arrow
<right>	right arrow
<pageup>	Page Up
<pagedown>	Page Down
<backspace>	Backspace
<delete>	Delete
<insert>	Insert
<enter>	Enter
<return>	Return
<home>	Home
<end>	End
<space>	Space bar
<f1>	function key 1
<f10>	function key 10

Figure 10: Special keys to which you can bind functions

These are some of the shortcuts I have bound keys to in order to facilitate easy navigation, but you can bind just about anything you like. See section B for a list of all the functions available to you when programming keys.

```
bind generic - first-entry
#minus sign takes me to the top of the list

bind generic = last-entry
```

```
#equals sign takes me to the bottom of the list

bind generic L bottom-page
#to the bottom,

bind generic M middle-page
#       middle, and

bind generic H top-page
#        top of the screen

bind index [ previous-unread
#to the previous unread message in the list

bind index ] next-unread
#or the next unread message.
```

The keys bound to ‘generic’ are configured for usage throughout the program. So above we have two key bindings that only function in the index, and in the pager while you’re reading a message, the [and] keys will have no effect.

4.4 Writing Macros

When you’d like a key stroke to execute something more complex than a simple command, use a macro to bind complex keystroke combinations to a key. Three examples will get you started. The first one assigns to the F2 key the command “change current folder to the inbox,” which you’d use if you were looking at the list of emails stored in another folder and wished to go back to the inbox. The second macro assigns to the F3 key the command “change current folder to the ‘Sent’ folder”. The third macro assigns to the key combination comma-c the command “save this message to the ‘Linux-list’ folder.” In each case, the backslash-n at the end is a carriage return. Far more complicated procedures are available to you in this way, using \r to hit carriage returns between events, and taking advantage of the long list of functions available to you (see section B for a complete list).

```
macro index <f2> "c !\n"
macro index <f3> "c =Sent\n"
macro index ,c "s=Linux-list\r"
```

4.5 Changing the Index View

We can set some other functionality that affects how the index view works. The following command instructs mutt to use an arrow to indicate which message is currently selected instead of an inverse video bar.

```
set arrow_cursor
#Changes the ‘bar’ cursor to an arrow.  Default is the bar.
```

4.6 Changing the Pager View

From the index, once you select a message, you're in the pager. By default, mutt uses its own pager, but if you prefer something else (like 'less'), you are welcome to use it. In the pager screen, the up and down arrow keys scroll the message up and down, but you can bind additional keys to control scrolling.

```
set pager="builtin"
#Use the built-in pager, not 'less'

set pager_context=5
#Paging down a message leaves 5 lines of overlap, so you don't get lost.

ignore *
# First, ignore all headers.
# In the next line we specify which ones we'd like to see.

unignore From To Reply-To Subject X-Mailer Cc Date
#Show these headers (ignoring all, then selecting some is easy)

hdr_order From: Date: To: Cc: Subject
#Order in which the headers are presented.
```

At the top of the screen is a small window previewing a few lines of the index, usually 3-5 messages before and after the message you're reading. If you'd like to change that, you can do so with the 'pager_index_lines' command. Finally, you can at the same time configure which headers you'd like to be shown by default (remember, you can see all of them by pressing the H key), and alter the order in which the headers are presented.

```
set pager_index_lines=4
#Show 2 messages on either side of the message I'm reading.
alternative_order text/enriched text/plain text/html
#In which order to show the body parts (I leave HTML for last)
```

Two additional variables are useful to set here. The first is 'tilde,' which, if set, pads the screen at the end of the message with tildes to indicate the end of file (the same way vim does). If that's interesting to you, include

```
set tilde
```

The other is 'resolve.' When you operate on a message, by default, Mutt then takes you to the next message automatically. But if the next message includes a big attachment you will be stuck until Mutt finishes downloading the entire attachment, which might take a while and be annoying if you are working online, i.e. with an IMAP mail account. You can prevent this by unsetting the 'resolve' variable, i.e.

```
unset resolve
```

Otherwise, redefine the delete command so that it takes you back to the index before deleting the message. That way the cursor will move automatically to the next message but not automatically open/download it. Add this to your .muttrc:[6]

```
folder-hook . 'macro pager d "<exit><delete-message>" "Delete the message"'
```

See section 4.10 for more information about folder hooks and how this folder hook works.

You can add lines like the following to ensure your pager works the way you think it will (up and down scrolling). I also added commands to go rapidly to the next message. Be creative.

```
bind pager <up> previous-line
bind pager <down> next-line
bind pager [ previous-unread
bind pager ] next-unread
```

4.7 Dealing with non-ASCII Character Sets

This is one of the more pesky configuration issues when using mutt – and many other console programs – and getting this part of your configuration will take the combination of several properly configured environments, including your console or xterm depending on which environment you use, the locale settings for your machine and user account, and even the fonts available to your system for displaying those characters. Your editor comes into play as well, as it must be capable of producing and properly encoding the non-Western characters you use. It's a big topic and merits a Woodnotes guide unto itself, and I'm working on one at the moment[7] so I'll put the basics here and refer you to sources better than this guide for more information.

Start with the mutt wiki article about foreign character sets, which is the most complete source of information on this topic that I know of (see sec. 5). It tells you how to set your $LOCALE variables in your bash shell configuration. That is, if you modify .bashrc and re-log in, you will have informed your system that on the console you like to work in a particular character set. More complex character sets (Russian, Polish) may additionally require ensuring the terminal or console has the font necessary to print those characters. Another good site of information on how to use Mutt in UTF-8 mode is available at Rano.org (see sec. 5).

You should, at the minimum, let mutt know what character set you *write* in. Consider this step optional if you are a native English speaker/writer, but if you write in any other language this step becomes more important. Add a line like the following to your .muttrc to let mutt know the order in which it should search for a character set with all the requested characters in it. It's an ordered list of character sets separated by colons.

```
set send_charset="iso-8859-1:utf-8"
```

[6] Thanks to Tim Brown for this trick.

[7] January 2007: visit `http://www.therandymon.com` for details.

4.8 Dealing With Particularly Troublesome Characters

Blame mutt for doing exactly what it's told to do. Character sets have very strict definitions of their character codes, and mutt takes those instructions literally, assuming other email software is telling the truth. But a lot of messages from Windows users either include Microsoft's extensions (like smart quotes and en-dashes) or include characters cut and pasted from other character sets but not reported. Some Microsoft email software, many web mail programs, and some news readers on various platforms report they are encoding characters in the ISO-8859-1 (Western Latin) character set but then proceed to use characters from the CP-1252 (Windows) character set. This causes you to read a lot extraneous characters in messages which Mutt presents to you in numerical form, like 221, 222, and 223. These numbers refer to octal codes for the characters, and appear when mutt doesn't know what to do with them. The following table shows some of the characters you are most likely to encounter:

Octal	Character
221	Left single quote
222	Right single quote
223	Left double quote
224	Right double quote
226	En dash

Figure 11: Special Character Octal Codes

A relatively clean way of dealing with this problem is to add a character set hook to your .muttrc, as follows. This instructs mutt to present characters using one character set when an email message reports that it has been encoded in another, as such: `charset-hook windows-1250 CP1250`. Repeat for CP1252 through CP1258, for example.

Alain Bench recommends you follow up with the iconv program (read about it in the man pages), which converts text from one character set to another.

Mutt includes a mechanism for substituting one character for another in what's called a display filter. Add the following to your muttrc file to declare characters you'd like to substitute:

```
set display_filter="tr '\\221\\222\\223\\224\\226'
'\\047\\047\\042\\042\\055'"
```

This is a command that passes those characters through the Unix 'tr' (translate) command, and I've had trouble with it on some mutt installations. Another recommendation is to install John Walker's demoronizer Perl script (see section 5)on your machine, and pass messages through the demoronizer as follows (for the record, I don't think this script works well if you are working in an UTF-8 encoded environment).

```
set display_filter="perl demoroniser.pl"
```

Take care to set the display filter to the actual path to where you've installed the demoroniser Perl script. I saved it in my home directory, so I had to use

```
set display_filter="perl ~/demoroniser.pl"
```

4.9 Advanced Printing: Mutt-Print

I very infrequently print email, but mutt-print is in my opinion a very worthwhile addition to the mutt software ecosphere and I now install it everywhere I use mutt. First download the mutt-print package (see section 5.2). You need to compile it yourself unless you're running a Debian derivative like Ubuntu, and in order to compile I had to download the iconv library and a Perl module, both of which were provided on my SUSE install disks, so they're by no means obscure software. Unpack the TGZ file and follow the instructions, and you will be fine. Mutt-Print uses LaTeX and Perl to format the files that are sent the printer, and they are gorgeous to look at, little penguin logo and all.

4.10 Hooks for Sending, Accounts, and More

Hooks are great ways to provide conditional functionality. They perform actions when certain criteria are met. For example, a set of folder hooks could say "list all my mailboxes in 'date received' order except the 'Office Jerks' folder, which I want listed by sender;" a send hook could say "whenever the recipient is 'Charlie' change my return address to the following; an account hook could say "whenever I'm looking at the imap://ISP-two.com account, my password is the following." That's all there is to it.

Hooks come in many flavors: folder-hooks, send-hooks, message-hooks, save-hook, mbox-hook, fcc-hook, and fcc-save-hook. I mostly use folder-hooks to choose a sorting pattern appropriate to the folder I'm looking at. Here are some examples, but if you think about the possibilities this functionality provides you'll see just how powerful they are.

```
# Folder Hooks
  folder-hook /var/spool/mail/randymon set sort=date-received
  folder-hook Avalon-list set sort=threads
  folder-hook '!' set sort=received

# Send Hooks
  send-hook '~t tgw@ISP-one.com$'
        'my_hdr From: Fat Penguin <eatmore@fish.com>'

  send-hook '~t @hotmail.com$'
       'my_hdr From: Tough Guy <name@address.com>'
```

5 Other Resources and More Information

5.1 Information and Tips

mutt and muttrc man pages: Type "man muttrc" for a glimpse at all the fabulous functionality available to you if you just take the time to configure it, and you'll see this Woodnotes Guide just scratches the surface.

The mutt home page: `www.mutt.org` A tremendous source of links to other resources, plus the wiki, sample config files, and more.

The mutt manual: `http://www.mutt.org/doc/manual/manual.html` The definitive guide to mutt.

Mutt Newsgroup: `comp.news.mutt` a very active usenet newsgroup, but watch your netiquette and do some Googling before asking simple questions.

The mutt-users mailing list: `http://www.mutt.org/mail-lists.html`

Mark's Mutt Fan and Tip Page: `http://mark.stosberg.com/Tech/mutt.html`

My First Mutt, by Bruno Postle: `http://mutt.blackfish.org.uk` A good start for many new users, and very user friendly

The Mutt Wiki: Character Sets `http://wiki.mutt.org` Good information on working with non-Latin character sets (Greek, Polish, Russian, etc.) light SMTP agents, and a lot more

Rano.org Unicode Mutt: `http://www.rano.org/mutt.html` info on getting mutt to work in a unicode environment

Postfix state of mind: `http://postfix.state-of-mind.de/patrick.koetter/smtpauth/smtp_auth_mailservers.html`: info on setting up Postfix to the point where you can send mail to your ISP.

Wolferman: A neat little script for keeping your mutt alias file and your abook addressfile in synch: `http://wolfermann.org/abook-autoexport`

Vincent Danen: Using Mutt on Mac OS X: `http://linsec.ca/Using_mutt_on_OS_X`

In addition, many people have posted their .muttrc files to the internet. Google for muttrc to find some of them.

5.2 Other Software with which Mutt Interfaces

The Little Brother Database: `http://www.spinnaker.de/lbdb/`

The Rolo project: `http://rolo.sourceforge.net`

Demoronizer: `http://www.fourmilab.ch/webtools/demoroniser/` or `http://freshmeat.net/projects/demoroniser/`

Mutt-print: `http://muttprint.sourceforge.net`

Putmail.py: `http://putmail.sourceforge.net`

mutt-ldap.pl: `http://www.bsdconsulting.no/tools/mutt-ldap.pl`

A Appendix: Patterns

```
~A              all messages
~b EXPR         messages which contain EXPR in the message body
~B EXPR         messages which contain EXPR in the whole message
~c USER         messages carbon-copied to USER
~C EXPR         message is either to: or cc: EXPR
~D              deleted messages
~d [MIN]-[MAX]  messages with ‘‘date-sent’’ in a Date range
~E              expired messages
~e EXPR         message which contains EXPR in the ‘‘Sender’’ field
~F              flagged messages
~f USER         messages originating from USER
~g              PGP signed messages
~G              PGP encrypted messages
~h EXPR         messages which contain EXPR in the message header
~k              message contains PGP key material
~i ID           message which match ID in the ‘‘Message-ID’’ field
~L EXPR         message is either originated or received by EXPR
~l              message is addressed to a known mailing list
~m [MIN]-[MAX]  message in the range MIN to MAX *)
~n [MIN]-[MAX]  messages with a score in the range MIN to MAX *)
~N              new messages
~O              old messages
~p              message is addressed to you (consults $alternates)
~P              message is from you (consults $alternates)
~Q              messages which have been replied to
~R              read messages
~r [MIN]-[MAX]  messages with ‘‘date-received’’ in a Date range
~S              superseded messages
~s SUBJECT      messages having SUBJECT in the ‘‘Subject’’ field.
~T              tagged messages
~t USER         messages addressed to USER
~U              unread messages
~v              message is part of a collapsed thread.
~x EXPR         messages which contain EXPR in the ‘References’ field
~y EXPR         messages which contain EXPR in the ‘X-Label’ field
~z [MIN]-[MAX]  messages with a size in the range MIN to MAX *)
~=              duplicated messages (see $duplicate_threads)
```

B Appendix: Functions available for Macros

B.1 Generic Functions

The generic menu is not a real menu, but specifies common functions (such as movement) available in all menus except for pager and editor. Changing settings for this menu will affect the default bindings for all menus (except as noted).

```
bottom-page                 L   move to the bottom of the page
current-bottom     not bound    move current entry to bottom of page
current-middle     not bound    move current entry to middle of page
current-top        not bound    move current entry to top of page
enter-command               :   enter a muttrc command
exit                        q   exit this menu
first-entry                 =   move to the first entry
half-down                   ]   scroll down 1/2 page
half-up                     [   scroll up 1/2 page
help                        ?   this screen
jump                   number   jump to an index number
last-entry                  *   move to the last entry
middle-page                 M   move to the middle of the page
next-entry                  j   move to the next entry
next-line                   >   scroll down one line
next-page                   z   move to the next page
previous-entry              k   move to the previous entry
previous-line               <   scroll up one line
previous-page               Z   move to the previous page
refresh                    ^L   clear and redraw the screen
search                      /   search for a regular expression
search-next                 n   search for next match
search-opposite    not bound    search for next match in opposite direction
search-reverse         ESC /    search backwards for a regular expression
select-entry             RET    select the current entry
shell-escape                !   run a program in a subshell
tag-entry                   t   toggle the tag on the current entry
tag-prefix                  ;   apply next command to tagged entries
top-page                    H   move to the top of the page
```

B.2 The Index

```
bounce-message              b   remail a message to another user
change-folder               c   open a different folder
change-folder-readonly ESC c   open a different folder in read only mode
check-traditional-pgp  ESC P   check for classic pgp
clear-flag                  W   clear a status flag from a message
```

```
copy-message                C   copy a message to a file/mailbox
create-alias                a   create an alias from a message sender
decode-copy               ESC C   decode a message and copy it to a file/mailbox
decode-save               ESC s   decode a message and save it to a file/mailbox
delete-message              d   delete the current entry
delete-pattern              D   delete messages matching a pattern
delete-subthread          ESC d   delete all messages in subthread
delete-thread               ^D   delete all messages in thread
display-address             @   display full address of sender
display-toggle-weed         h   display message and toggle header weeding
display-message           RET   display a message
edit                        e   edit the current message
edit-type                   ^E   edit the current message's Content-Type
exit                        x   exit without saving changes
extract-keys                ^K   extract PGP public keys
fetch-mail                  G   retrieve mail from POP server
flag-message                F   toggle a message's 'important' flag
forget-passphrase           ^F   wipe PGP passphrase from memory
forward-message             f   forward a message with comments
group-reply                 g   reply to all recipients
limit                       l   show only messages matching a pattern
list-reply                  L   reply to specified mailing list
mail                        m   compose a new mail message
mail-key                  ESC k   mail a PGP public key
next-new                  TAB   jump to the next new message
next-subthread            ESC n   jump to the next subthread
next-thread                 ^N   jump to the next thread
next-undeleted              j   move to the next undeleted message
next-unread           not bound   jump to the next unread message
parent-message              P   jump to parent message in thread
pipe-message                |   pipe message/attachment to a shell command
previous-new            ESC TAB   jump to the previous new message
previous-page               Z   move to the previous page
previous-subthread        ESC p   jump to previous subthread
previous-thread             ^P   jump to previous thread
previous-undeleted          k   move to the last undelete message
previous-unread       not bound   jump to the previous unread message
print-message               p   print the current entry
query                       Q   query external program for addresses
quit                        q   save changes to mailbox and quit
read-subthread            ESC r   mark the current subthread as read
read-thread                 ^R   mark the current thread as read
recall-message              R   recall a postponed message
reply                       r   reply to a message
resend-message            ESC e   resend message and preserve MIME structure
save-message                s   save message/attachment to a file
```

```
set-flag                    w   set a status flag on a message
show-version                V   show the Mutt version number and date
show-limit                ESC l   show currently active limit pattern, if any
sort-mailbox                o   sort messages
sort-reverse                O   sort messages in reverse order
sync-mailbox                $   save changes to mailbox
tag-pattern                 T   tag messages matching a pattern
tag-thread                ESC t   tag/untag all messages in the current thread
toggle-new                  N   toggle a message's 'new' flag
toggle-write                %   toggle whether the mailbox will be rewritten
undelete-message            u   undelete the current entry
undelete-pattern            U   undelete messages matching a pattern
undelete-subthread        ESC u   undelete all messages in subthread
undelete-thread             ^U  undelete all messages in thread
untag-pattern               ^T  untag messages matching a pattern
view-attachments             v  show MIME attachments
```

B.3 The Pager

```
bottom                not bound   jump to the bottom of the message
bounce-message              b   remail a message to another user
change-folder               c   open a different folder
change-folder-readonly    ESC c   open a different folder in read only mode
check-traditional-pgp     ESC P   check for classic pgp
copy-message                C   copy a message to a file/mailbox
create-alias                a   create an alias from a message sender
decode-copy               ESC C   decode a message and copy it to a file/mailbox
decode-save               ESC s   decode a message and save it to a file/mailbox
delete-message              d   delete the current entry
delete-subthread          ESC d   delete all messages in subthread
delete-thread               ^D  delete all messages in thread
display-address             @   display full address of sender
display-toggle-weed         h   display message and toggle header weeding
edit                        e   edit the current message
edit-type                   ^E  edit the current message's Content-Type
enter-command               :   enter a muttrc command
exit                        i   return to the main-menu
extract-keys                ^K  extract PGP public keys
flag-message                F   toggle a message's 'important' flag
forget-passphrase           ^F  wipe PGP passphrase from memory
forward-message             f   forward a message with comments
group-reply                 g   reply to all recipients
half-up               not bound   move up one-half page
half-down             not bound   move down one-half page
help                        ?   this screen
```

```
list-reply                L   reply to specified mailing list
mail                      m   compose a new mail message
mail-key              ESC k   mail a PGP public key
mark-as-new               N   toggle a message's 'new' flag
next-line               RET   scroll down one line
next-entry                J   move to the next entry
next-new                TAB   jump to the next new message
next-page                     move to the next page
next-subthread        ESC n   jump to the next subthread
next-thread              ^N   jump to the next thread
next-undeleted            j   move to the next undeleted message
next-unread       not bound   jump to the next unread message
parent-message            P   jump to parent message in thread
pipe-message              |   pipe message/attachment to a shell command
previous-line     BackSpace   scroll up one line
previous-entry            K   move to the previous entry
previous-new      not bound   jump to the previous new message
previous-page             -   move to the previous page
previous-subthread    ESC p   jump to previous subthread
previous-thread          ^P   jump to previous thread
previous-undeleted        k   move to the last undelete message
previous-unread   not bound   jump to the previous unread message
print-message             p   print the current entry
quit                      Q   save changes to mailbox and quit
read-subthread        ESC r   mark the current subthread as read
read-thread              ^R   mark the current thread as read
recall-message            R   recall a postponed message
redraw-screen            ^L   clear and redraw the screen
reply                     r   reply to a message
save-message              s   save message/attachment to a file
search                    /   search for a regular expression
search-next               n   search for next match
search-opposite   not bound   search for next match in opposite direction
search-reverse        ESC /   search backwards for a regular expression
search-toggle             \   toggle search pattern coloring
shell-escape              !   invoke a command in a subshell
show-version              V   show the Mutt version number and date
skip-quoted               S   skip beyond quoted text
sync-mailbox              $   save changes to mailbox
tag-message               t   tag a message
toggle-quoted             T   toggle display of quoted text
top                       ^   jump to the top of the message
undelete-message          u   undelete the current entry
undelete-subthread    ESC u   undelete all messages in subthread
undelete-thread          ^U   undelete all messages in thread
view-attachments          v   show MIME attachments
```

B.4 Aliases

```
search                      /   search for a regular expression
search-next                 n   search for next match
search-reverse          ESC /   search backwards for a regular expression
```

B.5 Queries

```
create-alias                a   create an alias from a message sender
mail                        m   compose a new mail message
query                       Q   query external program for addresses
query-append                A   append new query results to current results
search                      /   search for a regular expression
search-next                 n   search for next match
search-opposite    not bound    search for next match in opposite direction
search-reverse          ESC /   search backwards for a regular expression
```

B.6 The Attach menu

```
bounce-message              b   remail a message to another user
collapse-parts              v   toggle display of subparts
delete-entry                d   delete the current entry
display-toggle-weed         h   display message and toggle header weeding
edit-type                  ^E   edit the current entry's Content-Type
extract-keys               ^K   extract PGP public keys
forward-message             f   forward a message with comments
group-reply                 g   reply to all recipients
list-reply                  L   reply to specified mailing list
pipe-entry                  |   pipe message/attachment to a shell command
print-entry                 p   print the current entry
reply                       r   reply to a message
resend-message          ESC e   resend message and preserve MIME structure
save-entry                  s   save message/attachment to a file
undelete-entry              u   undelete the current entry
view-attach               RET   view attachment using mailcap entry if necessary
view-mailcap                m   force viewing of attachment using mailcap
view-text                   T   view attachment as text
```

B.7 The Compose menu

```
attach-file                 a   attach a file(s) to this message
attach-message              A   attach message(s) to this message
attach-key              ESC k   attach a PGP public key
```

```
copy-file                 C   save message/attachment to a file
detach-file               D   delete the current entry
display-toggle-weed       h   display message and toggle header weeding
edit-bcc                  b   edit the BCC list
edit-cc                   c   edit the CC list
edit-description          d   edit attachment description
edit-encoding            ^E   edit attachment transfer-encoding
edit-fcc                  f   enter a file to save a copy of this message in
edit-from               ESC f  edit the from: field
edit-file                ^X e  edit the file to be attached
edit-headers              E   edit the message with headers
edit                      e   edit the message
edit-mime                 m   edit attachment using mailcap entry
edit-reply-to             r   edit the Reply-To field
edit-subject              s   edit the subject of this message
edit-to                   t   edit the TO list
edit-type                ^T   edit attachment type
filter-entry              F   filter attachment through a shell command
forget-passphrase        ^F   wipe PGP passphrase from memory
ispell                    i   run ispell on the message
new-mime                  n   compose new attachment using mailcap entry
pgp-menu                  p   show PGP options
pipe-entry                |   pipe message/attachment to a shell command
postpone-message          P   save this message to send later
print-entry               l   print the current entry
rename-file               R   rename/move an attached file
send-message              y   send the message
toggle-unlink             u   toggle whether to delete file after sending it
view-attach             RET   view attachment using mailcap entry if necessary
write-fcc                 w   write the message to a folder
```

B.8 The Postpone menu

```
delete-entry              d   delete the current entry
undelete-entry            u   undelete the current entry
```

B.9 Browser

```
change-dir                c   change directories
check-new               TAB   check mailboxes for new mail
enter-mask                m   enter a file mask
search                    /   search for a regular expression
search-next               n   search for next match
search-reverse          ESC /   search backwards for a regular expression
```

```
select-new                N   select a new file in this directory
sort                      o   sort messages
sort-reverse              O   sort messages in reverse order
toggle-mailboxes        TAB   toggle whether to browse mailboxes or all files
view-file             SPACE   view file
subscribe                 s   subscribe to current mailbox (IMAP Only)
unsubscribe               u   unsubscribe to current mailbox (IMAP Only)
toggle-subscribed         T   toggle view all/subscribed mailboxes (IMAP Only)
```

B.10 PGP

```
view-name                 %   view the key’s user id
verify-key                c   verify a PGP public key
```

B.11 The Editor

```
backspace           BackSpace   delete the char in front of the cursor
backward-char             ^B   move the cursor one character to the left
backward-word          ESC b   move the cursor to the previous word
bol                       ^A   jump to the beginning of the line
buffy-cycle            Space   cycle among incoming mailboxes
capitalize-word        ESC c   uppercase the first character in the word
complete                TAB   complete filename or alias
complete-query            ^T   complete address with query
delete-char               ^D   delete the char under the cursor
downcase-word          ESC l   lowercase all characters in current word
eol                       ^E   jump to the end of the line
forward-char              ^F   move the cursor one character to the right
forward-word           ESC f   move the cursor to the next word
history-down       not bound   scroll down through the history list
history-up         not bound   scroll up through the history list
kill-eol                  ^K   delete chars from cursor to end of line
kill-eow               ESC d   delete chars from cursor to end of word
kill-line                 ^U   delete all chars on the line
kill-word                 ^W   delete the word in front of the cursor
quote-char                ^V   quote the next typed key
transpose-chars    not bound   transpose character under cursor with previous
upcase-word            ESC u   uppercase all characters in current word
```

C Appendix: Acknowledgments and License

This Woodnotes Guide is my own, in that the many weeks that went into researching, writing, editing, and correcting this document were mine; nevertheless I am indebted to the many sources of mutt information

I have taken and modified to learn about mutt myself and develop this Woodnotes Guide, including the archives of the mutt newsgroup and the personal mutt web pages of scores of mutt aficionados. They are too numerous to name individually, so instead I dedicate this document to them and release it to the public without charge. As I am a writer, not a programmer by trade, this is my contribution to the community in appreciation of the software that other volunteers have made available to me for free. Such is the Open Source community.

D Appendix: Version History

- 2 December 2009: offlineimap and concept of usage scenarios, wvText, querying an LDAP server, coloring the index, trimmed out some banter and discussion of mutt-ng, reorganization including appendices
- 3 March 2008: SMTP, forwarding attachments
- 14 April 2007: putmail.py, Mutt on Macs
- 6 March 2007: expanded abook and muttprint, more about pager view, attachments and 'resolve,' added IMAP header caching, default sending character set, and light smtp packages, some reorganization.
- 24 November 2006: major edits to And/Or tagging and flagging. Thanks to Chris Moore for spotting the errors. Also edits to fetchmail, postfix/sendmail, some capitalization fixes, and edits to the pager view.
- 18 November 2006: minor edits to Macros and Dealing with Word Docs
- 6 October 2006: Added licensing information and moved acknowledgments and version history to the top. Over 5000 people have already downloaded this guide!
- 13 May 2006: Updated some of the section about character sets, with help and suggestions from Alain Bench.
- 09 March 2006: Radically expanded version including major reorganization and appendices. This is more what I had in mind when I set out to write this guide. Page count approx. 37 pages.
- 08 September 2005: Initial version, written during a long sejour in Mozambique. *Muito obrigado!*