Great Email Features (You've Never Heard Of)

Find the power buried in your email program

By Jerry Peek

Your favorite email program can send and receive messages, handle folders, and do other basic things, but buried beneath its menus or keystroke combinations may be handy features you don't know about. And, because there are tens of email applications, it's also a safe bet that other email programs have tricks that yours doesn't. Email is a "killer app." Why not use a killer app, too?

In this extended "Power Tools" column, let's check out some great — and perhaps little known — features that at least a couple of popular email readers offer.

By the way, unless your program uses a proprietary folder format — like Microsoft Outlook, for instance — you may be able to switch between programs without having to "import" your folders into the new program. Stay tuned for some advice on testing new email programs.

Try the Plain Text View

MIME and HTML email can be nice, with fonts, graphics, links, and more, but "rich" email can also be full of clutter and can open the door to spammers. For instance, if a message has a link to an image at the spammer's server (even a tiny 1-by-1 pixel image that's basically invisible), the spammer will know that you got his message when your HTML email program asks his server for the tiny graphic file.

Many email programs have some protection against spammers' tricks, but another great way to avoid problems is using a plain-text, console-based email program like *Mutt* or *Pine*. You generally control these super-fast email programs with the keyboard instead of the mouse. Once you've learned their commands (or customized them to do exactly what you want), you can handle your mail in a flash.

For example, when you see a spam message, just tap the D key under your left middle finger and, bam, the spam is history. Or type something like SHIFT-CTRL-S to train your Bayesian spam filter with the message. The most flexible email programs let you add this kind of customization.

Another way to get the best of both worlds is by using *two* email programs. Start with the plain-text program to handle most of your mail. You'll spot spam easily, no matter how cleverly it's disguised. Then, if you have any MIME messages that must be displayed graphically, quit the plain-text program and start the graphical program. As long as both programs use the same mail folder format, there's a good chance that they'll work seamlessly. (This is especially true of MUAs that use the IMAP protocol to manage mail folders on a remote server and a protocol like LDAP for your address books.)

If you don't want to use a plain-text email program, you may be able to set your graphical program to default to a plain-text view. In Mozilla, for instance, choose the menu sequence View, Message Body As, Plain Text. (This probably won't show the complete message, though. If you want that, use the "View message source" command.)

To learn more about what you might see in the plain text view, read the sidebar "Behind the MIME Curtain."

Delayed "Read" Marking

In general, as soon as you open an email message, your mail program marks the message "read." For example, a message summary in the folder view may change from boldface (for unread messages) to plain text.

Do you like to scan through your mail, handling some messages right away and leaving others for later? *Evolution* and *KMail* are two mailers that let you choose to mark a message as "read" only after a certain amount of time has passed. For instance, if you set this option to 5 seconds in KMail (Settings, Configure KMail, Folders), the message won't be marked read unless you look at it for more than five seconds.

This is great when scanning through a folder, looking for messages you need to deal with immediately. Set the delay to a few seconds and, if you haven't deleted the message or moved onto another one by that time, the message will be marked as read; otherwise, it'll stay marked "unread" so you can find it later.

Redirecting Mail

You've probably forwarded an email message before. To forward a message, you compose a new message that includes the message you want to forward, either in that message body or as an attachment.

Figure One shows a message being forwarded. Because the original message header is now in the body of another message, the recipient's email folder summary probably won't show who the original is from or when it was sent.

So, if the recipient (*zoe@d.com*) wants to find the original message, she can't search for messages from the original sender (*jo@a.com*). Instead, she needs to remember that Al forwarded the message to her, or has to search the message bodies for Jo's email address. If she wants to reply to the original sender, she needs to compose a new message (or, possibly, *burst*, or unpackage, the forwarded message into a new file and reply to it from there).

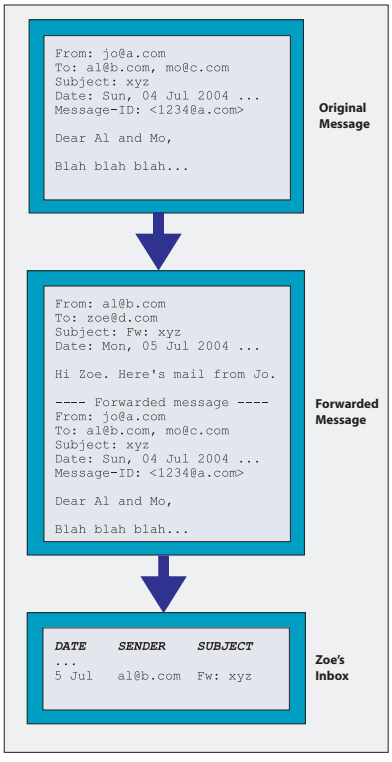

FIGURE ONE: Forwarding a message

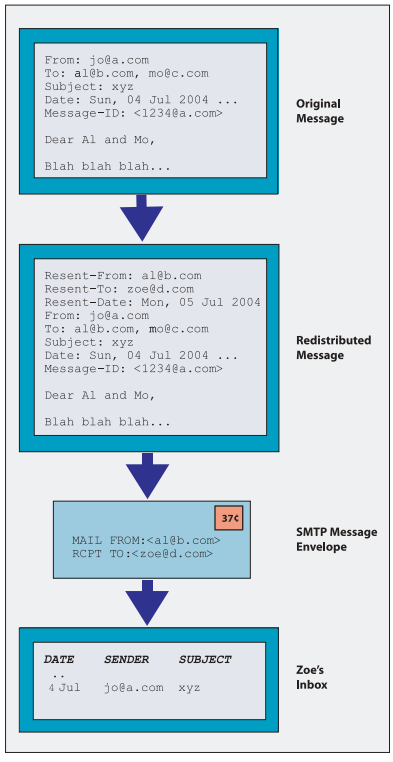

FIGURE TWO: Redistributing a message

Some email programs let you forward a message in another way, often called *redistribution* or *redirection*. Instead of burying the original message header in the forwarded message body, a message is re-sent with its original header in place. New header fields — typically `Resent-to`, `Resent-From`, and `Resent-Date` — are added. The recipient (here, *zoe@d.com*) gets the message in the same way as if the original sender (*jo@a.com*) had sent it to her.

The recipient of a redirected message can then reply to the original sender using a typical reply command. She can also search for the message by the original sender's name or address, and so on. (This is great when you want to include a bunch of new recipients in a discussion. Just redistribute a message to these new recipients, and they can all reply directly to the original sender.) *Figure Two* shows how this works.

What's happening here? Redistribution uses a fundamental email feature that's often misunderstood: the actual email address of a message sender and recipient isn't necessarily in the message header! It's in the *message envelope*. You can get details from the July 2003 feature "Personal Post," available online at http://www.linux-mag.com/2003-07/email_02.html.

When a user's MUA (*mail user agent*, the email program) redistributes a message, it can pass the recipient addresses to the system MTA (mail transfer agent, like sendmail or Exim). In redistribution, the original `From` and `To`, and other addresses in the header are ignored.

This seems handy — and it can be. Why isn't it used more often? Because...

- The person re-sending the message can't add a note above it, edit the contents, or modify the message in any way. (There's a one-to-one mapping between the original `Message-ID` field and the contents of the original message. If you modify the message contents, you should give it a new `Message-ID`.)
- If the recipient sorts her inbox by the date the message was sent, an old message that's redistributed to her can be placed far back in her message list. For example, if today is July 11 and Al redistributes a message to Zoe that was sent July 4, she'll have to look back a week in her inbox to find the message.
- If the recipient has mail filtering by sender (the `From` header field), but her filter isn't set to accept mail from the original sender, the mail may be rejected or put in a folder where she might not find it.

If the recipient might have these problems, one easy workaround is simply to tell her that the message is coming. If you use

redistribution often, recipients can set their filters to correctly handle mail with a `Resent-From` field.

Different email programs have different names for this feature. In KMail, it's the menu option Message, Forward... Redirect. In *Sylpheed* and Evolution, it's simply called "redirect." *MH* calls it "dist."

If your email program can't redirect mail, you can probably do it manually. Simply save the message in a file and pass it directly to your MTA. For details, see the supplement to this article at http://www.linux-mag.com/downloads/2004-07/power/forward.html.

Send Yourself a Reminder

If your job is email-driven, writing a message to yourself can be a great reminder.

This is easy with Evolution's Actions, Post New Message menu option. It lets you choose the destination folder, then stores whatever you type as a new message in that folder.

> With a text-based mail reader, you'll spot spam easily, no matter how cleverly it's been disguised

Of course, you can also just compose a new message to yourself. From a terminal window, the simple *mail* utility can send text from the keyboard, the output of a program, or the contents of a file. Here are three examples:

```
$ mail -s "do at work" \
  joe@w.xyz < to_do.txt
$ mail -s "Fix the ABC report" joe
The end of page 3 needs a summary.
^D
$ make prog | mail -s "Fix this" joe
```

The first example mails text file `to_do.txt` to `joe@w.xyz`. In the second example, *mail* reads from the keyboard until you type CONTROL-D at the start of a new line, then the text you typed is emailed in the message body. The third example pipes the standard output of command `make prog` into an email message with the subject `Fix this`. If you want to mail the standard error, too, use one of the two versions below. The first is for *tcsh* and the second is for Bourne-type shells:

```
% make prog |& mail -s "Fix this" joe

$ make prog 2>&1 | mail -s "Fix this joe
```

One more note: The *mail* utility typically assumes that you have a *mail transfer agent* (MTA) like *sendmail* running on your local system. If this isn't true — or if you just like simple command-line interfaces to mail — you might check out *nail*, a MIME-capable *mail* replacement by Gunnar Ritter, available at http://nail.sourceforge.net. You can specify a remote SMTP server with the *nail* option `smtp`.

Sharing Mail

One of the best features of Linux and Unix-based systems has always been that they're *multi-user*: the systems are designed to be used by more than one person at a time. So, it's easy for a group of people to share files. (There's more information in the November 2002 "Power Tools" column "Sharing Files Carefully," also available online at http://www.linux-mag.com/2002-11/ power_01.html.)

Many Unix and Linux email folders are stored in the *mbox* format, where each folder is a single file that stores multiple messages. In the MH format, each folder is a separate Linux directory, and each message is stored in a separate file. (These formats are described in the July 2003 feature "Personal Post.") A third format, *maildir*, uses a set of subdirectories to avoid problems with locking and message corruption.

One of the advantages of MH format is that — thanks to the Unix permission scheme — it's easy to share individual messages and even entire folders, with other users.

This can be tremendously useful in groups that need to see the same email. For example, as in a bug-tracking system, where new bugs come in via email, there's no need to send tens or hundreds of copies of the same message. Everyone can look in the same folder, reading and handling whatever messages are there.

Traditional email programs assume that all mail folders are within a single directory on your account. Programs that are fully MH-compatible can track folders anywhere on the filesystem, opening any folder or message that the current user has permission for. Unfortunately, many email programs that use MH storage don't allow this.

For instance, Sylpheed doesn't support MH *sequences*, which let each user keep track of the message they're reading, their unread messages, and so on. Evolution resets the permissions on an MH message file to make the message readable only by its owner. *MH-E*, which runs within the *Emacs* editor, is one email program that can handle MH folders anywhere. See http://mh-e.sourceforge.net. And, of course, you can also use the standard MH utilities — now called *nmh* — to handle shared folders. See http://www.nongnu.org/nmh and the online MH book at http://www.ics.uci.edu/~mh/book/mh/shafol.htm.

Mulberry, from http://www.cyrusoft.com (which started at Carnegie-Mellon University's *Cyrus* Project), takes advantage of IMAP features to allow sharing.

Message Priorities and Scoring

People who send you a message may assign a priority to it. This is done through the header field `X-Priority`, which has a value from 1 (the highest priority) to 5 (the lowest). (Microsoft has a proprietary header field `X-MSMail-Priority`, though their products also seem to set `X-Priority`.)

Another common header field is `Precedence`. Bulk messages — ones from mailing lists, for instance — could have the header line `Precedence: bulk`. Considerate senders may also set `Precedence: junk`. These values aren't well-standardized, but they may be useful in message filters (as you'll see).

Your mail program might flag these messages automatically — with a mark in the folder listing, for example. Some email programs also let you manually assign a priority or a classification to messages in your folders. For instance, in *Mozilla*, the menu option Message, Label, Important adds a red highlight to the folder summary of a message, and Message, Label, Work makes the summary line orange.

One of the most flexible priority schemes comes with Mutt, the highly-configurable text-based MUA. It's called *scoring*. (Although we're trying not to focus on specific email programs in this article, Mutt has so many unique features like scoring that it's worth looking at it to give you a feel for what a really sophisticated email program can do.)

To score your email, you write a set of rules that assign numeric scores to messages based on criteria such as what's in the header, whether the message has been marked as deleted, and many others. Then you can sort a folder based on message scores. Messages with the highest score appear first in the folder.

A score can be a fixed number. For instance, a message with "Viagra" in the subject is assigned a score of 0, and a message from someone important gets a high score like 1000.

Let's see configuration lines to do that. Each scoring line uses the Mutt patterns, which start with a tilde (`~`) character. So, for example, `~s` matches the message subject and `~D` matches a message that has been marked for deletion:

```
score '~s viagra'  =0      # spam
score '~f streicher' =1000 # my editor
score '~D'      =0         # deleted
```

A score can also be calculated based on a series of scoring lines. For instance, you could start all messages (the Mutt pattern `~A`) with a default score of 500. Later rules raise the score if the message is from a team member or has been flagged, and lower the score if the message is from a person who sends junk:

```
score '~A'           +500          # default
score '~f Ed.Doe|~f Zoe.Lee' +200 # my team
score '~F'           +100          # flagged
score '~f jpeek'       -500        # boring
```

Then, to sort your folders by score (highest first), use:

```
set sort=score
```

It might be convenient to define some keyboard macros — one to sort the folder by score, another to sort by date, and so on. For more information, see the *Mutt User's Guide*'s section called "Patterns."

Filtering on Custom Header Fields

Most modern email programs have filtering and virtual folders that let you sort and organize messages by their content — for instance, by the person who sent the message.

If your program is flexible enough, it likely lets you match any header field — not just common fields like `From` or `Subject`. To set this up, it's worth looking at the complete message header to see whether you can find a field that's unique to the message. For example, messages from a particular mailing list may have a `X-Mailing-List` field.

Your email program may do some of this analysis for you. For instance, when you use the KMail command Tools, Create Filter... on a mailing list, it tries to find a filter that uniquely identifies mail from that list. (It puts the guessed list name on the *Mailing-List* submenu entry.)

Save Large Mail for Later

If you download mail from a remote POP server, you may not want to retrieve some really large messages at some times. For instance, if you're on the road with your laptop and you only have a dialup net connection, maybe you don't want to download that two megabyte message with photos of your nephew. Find out whether your email program has message size limits.

In Mozilla and *Netscape*, for instance, you can set a size limit for a particular account under the Disk Space preferences category. Check the box "Do not download messages that are larger than..." and choose a maximum size. When you get a large message, you'll see only a "stub" in your folder, showing you the first few lines of the message and giving you a link to click to download all of it. If you don't click the link on these stub messages, the entire message waits on the POP server until you download it from somewhere else.

In KMail, set a download filter for large mail to "Download mail later." Then you can retrieve the message manually, later, or, for example, from another email program at home.

What Else?

There are many, many email programs available. If your cur-

rent program doesn't do what you want, or you suspect that more powerful programs might save you time, it's worth browsing a list — on freshmeat.net, for example — to see what's out there.

Once you find a program you'd like to try, you might set your mail system to forward duplicates of some or all of your messages to an alternate account or into an alternate group of folders (under a different top-level directory), and use those to experiment with other email programs.

In *procmail*, the `c` flag makes a recipe that copies. If your system uses a home-directory *.forward* file, insert the other address where you want a copy as well as your current username (mailbox name) with a backslash (\) before it, like this:

```
jpeek@somewhere-else.xyz, \jpeek
```

To get other users' opinions of the program, try a Google search, read message boards, and so on.

Play around with the program to be sure it's solid. (There's nothing worse than losing all your email to a program crash!)

If you switch to the new program, consider running a *cron* job to back up your mail folders periodically (with *tar*, for instance) until you've had enough experience to feel confident.

If email is a big part of your life, finding an email program that does what you need can save time and headaches.

Jerry Peek is a freelance writer and instructor who has used Unix and Linux for over 20 years. He's happy to hear from readers at http://www.jpeek.com/contact.html.

BEHIND THE MIME CURTAIN

What's in a message? Years ago, before the MIME standard, a message was generally plain text: a header and a body. After MIME was designed, users could send data seamlessly through the mail — including, unfortunately, infected attachments, worms, and other mischief that leaky mail programs could automatically interpret.

Spammers use all sorts of tricks to make their messages unique to keep spam filters from recognizing the message. For example, they might add hundreds of bogus HTML tags or long series of random words inside comment strings. Most email programs hide all of this from you. If you read a message without MIME processing, you'll spot this mischief immediately.

Many email messages also contain HTML markup: fonts, links, and more. The MIME standard allows a single message to contain more than one format: for instance, a plain-text version and an HTML version. The simplest version (which is usually plain text) should come first in the message body.

So, with plain-text email programs that ignore the MIME coding, you'll see the plain-text part of a message first. (This is just what you want.)

Let's look at a MIME `multipart/alternative` message, as you might see it in a plain-text view:

```
From: Laundry News <do-not-reply@wash.xyz>
To: Our Hard-Washin' Friends <list@wash.xyz>
Subject: Laundry News, July 2004
MIME-Version: 1.0
Content-Type: multipart/alternative;
   boundary="mime_separator_12345"

This message is in MIME format. Since you
are seeing this text, your email program
probably doesn't understand MIME.

—mime_separator_12345
Content-Type: text/plain; charset=us-ascii

Dear Friends,

Welcome to the Laundry News!

Keep scrubbin'!

—mime_separator_12345
Content-type: text/html

<P>
Dear Friends,
<P>
<I>Welcome</I> to the Laundry News!
<P>
<B>Keep scrubbin'!</B>

—mime_separator_12345—
```

A MIME *multipart* message starts with a `Content-Type` header field. For multipart content, this field also declares the *boundary* string used to separate the message parts. Each instance of the boundary starts with two dashes (–) and the last instance also ends with two dashes. Outside the boundaries, you can put other text. MIME-capable MUAs won't display this out-of-bounds text, but plain-text MUAs will. This is one place spammers put random words or other garbage to foil pattern-matching anti-spam software.

Plain-text email programs also let you see how attachments are coded and the file types that were assigned. Worm writers typically disguise executable attachments.

For instance, here's the start of a body part which might appear like a document but is actually executable:

```
—_NextPart_000_0016
Content-Type: application/octet-stream;
   name="document.doc
                .exe"
```

You'll learn tricks like these if you use a plain-text email program (for the first pass through your messages, at least, as we explained in the body of this article).