1 Adding Diverse Multimedia Format Support to Established RFC822 Mail and Bulletin Board Readers

Nathaniel S. Borenstein
<nsb@thumper.bellcore.com>
Bellcore

Abstract

It is surprisingly easy to use the RFC1049 "Content-type" header to turn virtually *any* mail reading interface into a multi-media mail reading interface. Mail readers are simply modified to use the new "metamail" program whenever they receive non-text mail. The metamail program is itself easily customizable by the use of a "mailcap" file that specifies the media types supported by a given site or user. Given the existence of the metamail program, this document explains how to add multimedia support to sixteen very different mail reading programs, including all of the most popular UNIX mail reading programs.

Motivation

Multimedia mail has been a long time coming. The biggest impediments to multimedia mail have been the absence of standards and the pain users experience when they convert to a new mail system. Thus, for example, to receive the benefits of the rich multimedia capabilities of the Andrew mail format, you have, in the past, had to convert from whatever other mailer you're using to one of the Andrew-based mailers.

In the context of a new research project (the MAGICMAIL language for active messaging) the author of this document has discovered that it is remarkably easy to extend nearly any mail reading interface so that it recognizes certain "Content-type" headers and, when it receives mail with such headers, calls an appropriate external program to display or interpret the mail body. Because this functionality seems so generally useful, I have taken the time to create this document, in the hope that it will help to make various kinds of multimedia mail functionality more widely available.

The ideas and code in this system apply equally well to both mail and bulletin board programs. Although this document talks mostly about mail, it applies equally to bulletin board readers, and in fact one bulletin board reading program (the Berkeley "msgs" program) is included in the set of programs discussed here.

The Basic Idea

Basically, there are only two things you have to do to each mail reading program:

1. Make the mail reader notice the special header ("Content-type") that marks a message as a non-text message. (In the case of mail readers that already understand certain content-types, such as Andrew, the mail reader must be modified only to deal with the content-types it does *not* already know how to handle.

2. When the special header appears, instead of (or, if it's much easier, in addition to) showing the user the body of the message, the mail reader must send that body off to the metamail interpreter. The metamail interpreter includes features that deal with the diverse situations of terminal-oriented and window-oriented mail readers.

Beyond this, of course, you have to make sure that the appropriate interpreters are available on your users' search paths. In particular, you'll need the metamail binary, an appropriately-configured mailcap file, and interpreters for whatever mail formats you support locally.

For information on how to use the metamail program or how to customize a mailcap file, see the metamail program's manual entry. This document will describe the specific patches that can be used to make certain established mail-reading programs work with metamail.

A Variety of Mail and Bulletin Board Reading Interfaces

With this document, you can patch all of your site's mail reading interfaces to support whatever multimedia formats are deemed useful at your site. This means that those who regularly use the multimedia tools can begin to send mail in those formats freely, without worrying about the ability of any local user to interpret the mail. It is my intent to make this document exhaustive; as time goes on, I hope it will grow to include an ever widening set of mail reading interfaces. Currently it includes all of the mail reading interfaces that I know to be in use anywhere in Bellcore's research laboratories.

Currently this document describes how to add support for the following mail readers:

Berkeley Mail (/usr/ucb/Mail, /usr/ucb/mail, and Tahoe mail)
SunMail (another version of Berkeley mail, but rather different)
Xmail (an X11 interface to Berkeley mail)
Mailtool (a SunTools interface to Berkeley mail)
Imail (Bellcore MICE mailer)
PCS readmail/rdmail/sreadmail (another Bellcore mailer)
MH -- Rand Message Handling System
XMH -- X11 Interface to Rand Message Handling System
Rmail -- GNU Emacs mail reading package
VM -- Another GNU Emacs mail reading package
CUI -- Andrew low-end mail reader
VUI -- Andrew termcap-based mail reader
Messages -- Andrew multimedia mail reader
Elm -- Mail reader from HP.
Msgs -- simple Berkeley bulletin board reader

If you have mail readers that are not dicussed here, you will still probably find some of this code useful as a model. If you develop a patch for some other mail reader, and you send it back to me, I'll include it in future versions of this document.

NOTE: For the programs that were written in C, all of the patches are surrounded by "#ifndef NOMETAMAIL/#endif". This was done so that a single source could easily be maintained even for use at those sites that for some reason desire to inhibit the metamail functionality. Such sites need only compile with "-DNOMETAMAIL" to have the modified mailers behave exactly like their unmodified predecessors.

ADDITIONAL NOTE: Most of these patches send a message to metamail if it has ANY content-type header. Technically, this is not necessary with a content-type such as

However, this can't just be a literal string comparison -- the full MIME content-type syntax must be parsed and the "charset" value checked. For example, you WOULD want to pass the following to metamail:

Content-type: text/plain; charset=iso-8859-8

However, there is not really a need to pass the following to metamail:

Content-type: text/plain; something-else=foobar; charset=us-ascii

Instead of adding the content-type parsing to each mailer, most of the patches below simply call metamail for any message with a content-type header. This is a bit inefficient for plain ascii text, but most such messages probably won’t have content-type headers anyway, and it makes the patches much simpler. Programmers incorporating these patches into "production" versions of mail readers to be released to the world might consider parsing the content-type header in full.

1.1 Berkeley Mail (/usr/ucb/[Mm]ail, and Tahoe mail)

NOTE: If you don’t have the sources for Berkeley mail, you can get about 95% of metamail functionality to work by setting your PAGER environment to "metamail" and reading everything with the pager (e.g. by using "more" instead of "type".) However, things will work slightly better if you can modify the sources.

There are several versions of Berkeley mail; this describes the patch to the versions I happened to have.

All of the changes are localized to the file cmd1.c However, the way that the Mail program handles output piped to "more" (or some other paging program) makes the patch about twice as complicated as it otherwise would be.

At any rate, there are four changes:

1. In cmd1.c, there is a routine called "type1" -- in my version it starts on line 321. At the end of the declarations at the top of this routine, there is a line that says:

```
        FILE *ibuf, *obuf;
```

(In some versions, the "*ibuf" is omitted. That’s fine.) Immediately after that line, **add the following declaration:**

```
#ifndef NOMETAMAIL
        int PipeToMore = 0;
#endif
```

2. In the same routine, about 18 lines down from the previous patch, there is a code fragment that says:

```
                        cp = value("PAGER");
                        if (cp == NULL || *cp == '\0')
                                cp = MORE;
                        obuf = popen(cp, "w");
                        if (obuf == NULL) {
                                perror(cp);
                                obuf = stdout;
                        }
                        else {
                                pipef = obuf;
                                sigset(SIGPIPE, brokpipe);
                        }
```

You need to **put three new lines before this code fragment and one new line after it**. The end result should look like this:

```
#ifndef NOMETAMAIL
                   PipeToMore = 1;
#else
```

```
                        cp = value("PAGER");
                        if (cp == NULL || *cp == '\0')
                                cp = MORE;
                        obuf = popen(cp, "w");
                        if (obuf == NULL) {
                                perror(cp);
                                obuf = stdout;
                        }
                        else {
                                pipef = obuf;
                                sigset(SIGPIPE, brokpipe);
                        }
#endif
```

Note that in some versions of UNIX, the code will say "signal" instead of "sigset" -- you should use whichever is used in your original. In some other variants (notably Ultrix) this code is significantly different. What matters is that the code that sets up the pager be preceded by the #ifndef ... #else lines, and followed by the #endif line.

3. About 8 lines further down in that routine, you'll find something that looks like this:

```
        for (ip = msgvec; *ip && ip-msgvec < msgCount; ip++) {
                mesg = *ip;
                touch(mesg);
                mp = &message[mesg-1];
                dot = mp;
                print(mp, obuf, doign);
```

In the Tahoe version, there is another line before the "print" line:

```
                if (value("quiet") == NOSTR)
                        fprintf(obuf, "Message %d:\n", mesg);
```

The "print" line is, in my version of Berkeley mail, line 367, and in my version of Tahoe mail, line 343. **REPLACE the "print" line with the following code**:

```
#ifndef NOMETAMAIL
#include <sgtty.h>
                if (!getenv("NOMETAMAIL") && nontext(mp)) {
                    char Fname[100], Cmd[120];
                    FILE *fp;
                    int code;
                    struct sgttyb ttystatein, ttystateout;

                    sprintf(Fname, "/tmp/mail-metamail.%d.%d", getpid(), getuid());
                    fp = fopen(Fname, "w");
                    if (!fp) {
                        perror(Fname);
                    } else {
                        send(mp, fp, 0);
                        fclose(fp);
                        sprintf(Cmd, "metamail -z %s -m Mail %s", (PipeToMore) ? "-p" : "",
Fname);
                        if (obuf != stdout) {
                            pipef = NULL;
                            pclose(obuf);
                            obuf = stdout;
                        }
                        gtty(fileno(stdin), &ttystatein);
```

```
                        gtty(fileno(stdout), &ttystateout);
                        code = system(Cmd);
                        stty(fileno(stdin), &ttystatein);
                        stty(fileno(stdout), &ttystateout);
/*                      The following line would cause the raw mail to print out if metamail
failed */
/*                      if (code) print(mp, obuf, doign); */
                    }
                } else {
                    if (PipeToMore && stdout == obuf) {
                        obuf = popen(MORE, "w");
                        if (obuf == NULL) {
                            perror(MORE);
                            obuf = stdout;
                        }
                        else {
                            files[fileno(obuf)].filep = obuf;
                            files[fileno(obuf)].flags = PIPE_OPEN & ~KEEP_OPEN;
                            pipef = obuf;
                            sigset(SIGPIPE, brokpipe);
                        }
                    }
                    print(mp, obuf, doign);
                }
#else
                print(mp, obuf, doign);
#endif
```

Note that in some versions, it will say "send" instead of "print" -- use whichever routine name is used in your version. Note also that on some versions of UNIX, you will need to use "signal" instead of "sigset". Finally, in some versions of the sources, the "files" array does not exist, and the two lines that refer to it can simply be eliminated.

4. At the very end of cmd1.c, **add the following new routine**:

```
#ifndef NOMETAMAIL
#include <ctype.h>

nontext(mp)
struct message *mp;
{
    long c;
    FILE *ibuf;
    char line[LINESIZE], *s, *t;

    ibuf = setinput(mp);
    c = mp->m_size;
    while (c > 0L) {
        fgets(line, LINESIZE, ibuf);
        c -= (long) strlen(line);
        if (line[0] == '\n') return(0);
        for (s=line; *s; ++s) if (isupper(*s)) *s = *s - 'A' + 'a';
        if (!strncmp(line, "content-type:", 13)) return(1);
    }
    return(0);
}
#endif
```

These two changes should be all you need to do in order to make Berkeley mail work with metamail, assuming that you already have the "metamail" binary installed somewhere on your search path.

1.2 Sun Mail (another version of Berkeley mail)

Sun's version of Berkeley mail is sufficiently different from the others to warrant a separate section. Patching the "Mail" program properly will *almost* automatically add metamail support to mailtool and xmail, because they simply call Mail. However, some further changes are necessary for those programs, as noted in subsequent sections.

All of the changes are localized to two files, cmd1.c and cmd2.c In particular, there are four changes:

1. In cmd1.c, there is a routine called "type1" -- in my version it starts on line 317. At the end of the declarations at the top of this routine, there is a pair of lines that say:

```
        FILE *ibuf, *obuf;
        void (*saveint)();
```

Immediately after that line, **add the following declaration:**

```
#ifndef NOMETAMAIL
        int PipeToMore = 0;
#endif
```

2. In the same routine, about 18 lines down from the previous patch, there is a code fragment that says:

```
                        obuf = popen(MORE, "w");
                        if (obuf == NULL) {
                                perror(MORE);
                                obuf = stdout;
                        }
                        else {
                                pipef = obuf;
                                sigset(SIGPIPE, brokpipe);
                        }
```

You need to **put three new lines before this code fragment and one new line after it**. The end result should look like this:

```
#ifndef NOMETAMAIL
                    PipeToMore = 1;
#else
                        obuf = popen(MORE, "w");
                        if (obuf == NULL) {
                                perror(MORE);
                                obuf = stdout;
                        }
                        else {
                                pipef = obuf;
                                sigset(SIGPIPE, brokpipe);
                        }
#endif
```

3. About 8 lines further down in that routine, you'll find something that looks like this:

```
        for (ip = msgvec; *ip && ip-msgvec < msgCount; ip++) {
                mesg = *ip;
                touch(mesg);
                mp = &message[mesg-1];
                dot = mp;
                print(mp, obuf, doign);
```

The "print;" line is, in my version, line 363. **REPLACE the "print" line with the following code**:

```
#ifndef NOMETAMAIL
#include <sgtty.h>
                if (!getenv("NOMETAMAIL") && nontext(mp)) {
                    char Fname[100], Cmd[120];
                    FILE *fp;
                    int code;
                    struct sgttyb ttystatein, ttystateout;

                    sprintf(Fname, "/tmp/mail-metamail.%d.%d", getpid(), getuid());
                    fp = fopen(Fname, "w");
                    if (!fp) {
                        perror(Fname);
                    } else {
                        msend(mp, fp, 0, fputs);
                        fclose(fp);
                        sprintf(Cmd, "metamail -z %s -m Mail %s", (PipeToMore) ? "-p" : "",
Fname);
                        if (obuf != stdout) {
                            pipef = NULL;
                            pclose(obuf);
                            obuf = stdout;
                        }
                        gtty(fileno(stdin), &ttystatein);
                        gtty(fileno(stdout), &ttystateout);
                        code = system(Cmd);
                        stty(fileno(stdin), &ttystatein);
                        stty(fileno(stdout), &ttystateout);
/*                      if (code) print(mp, obuf, doign); */
                    }
                } else {
                    if (PipeToMore && obuf == stdout) {
                        obuf = popen(MORE, "w");
                        if (obuf == NULL) {
                            perror(MORE);
                            obuf = stdout;
                        }
                        else {
                            pipef = obuf;
                            sigset(SIGPIPE, brokpipe);
                        }
                    }
                    print(mp, obuf, doign);
                }
#else
                    print(mp, obuf, doign);
#endif
```

2. At the very end of cmd1.c, **add the following new routine**:

```
#ifndef NOMETAMAIL
```

```
#include <ctype.h>

nontext(mp)
struct message *mp;
{
    long c;
    FILE *ibuf;
    char line[LINESIZE], *s, *t;

    ibuf = setinput(mp);
    c = mp->m_size;
    while (c > 0L) {
        fgets(line, LINESIZE, ibuf);
        c -= (long) strlen(line);
        if (line[0] == '\n') return(0);
        for (s=line; *s; ++s) if (isupper(*s)) *s = *s - 'A' + 'a';
        if (!strncmp(line, "content-type:", 13)) return(1);
    }
    return(0);
}
#endif
```

3. **This step is required only if you want the changes to work right with Sun's MailTool program.** In cmd2.c, there is a routine called "savemsglist" -- in my version it starts on line 187. About a page down in that routine, you'll find a line that looks like this:

```
               mp = &message[mesg-1];
```

This line is, in my version, line 215. Immediately following that line (i.e. between line 215 and line 216, in my version) **add the following code**:

```
#ifndef NOMETAMAIL
#include <sgtty.h>
               if (!mark && !getenv("NOMETAMAIL") && nontext(mp)) {
                   char Fname[100], Cmd[120];
                   FILE *fp;
                   int code;
                   struct sgttyb ttystatein, ttystateout;

                   sprintf(Fname, "/tmp/mail-metamail.%d.%d", getpid(), getuid());
                   fp = fopen(Fname, "w");
                   if (!fp) {
                       perror(Fname);
                   } else {
                       msend(mp, fp, 0, fputs);
                       fclose(fp);
                       fclose(obuf); /* To allow appending, sigh */
                       sprintf(Cmd, "metamail -z -m Mail %s >> %s", Fname, file);
                       gtty(fileno(stdin), &ttystatein);
                       gtty(fileno(stdout), &ttystateout);
                       code = system(Cmd); /* ignore it if it fails */
                       stty(fileno(stdin), &ttystatein);
                       stty(fileno(stdout), &ttystateout);
                       if ((obuf = fopen(file, "a")) == NULL) {
                           perror("");
                           return;
                       }
                       continue;
```

```
                        }
                }
#endif
```

These three changes should be all you need to do in order to make Sun’s version of Berkeley mail work with metamail, assuming that you already have the metamail binary installed somewhere on your search path.

1.3 Xmail (X11 Interface to Berkeley Mail)

Xmail is a program that provides a graphical interface to Berkeley mail. It does most of its work by talking to the Berkeley mail program itself. Thus, if you want Xmail to work with metamail, you have to first of all upgrade your version of the Berkeley Mail program as described in the previous sections.

Once you have done this, everything would work fine automatically except that there's no way for metamail to tell that it has been called non-interactively. You can tell it that this is the case (so that it won't try to ask any questions) by setting the MM_NOTTTY environment variable. Thus, a simple way to fix xmail would be to rename the "xmail" program to be "xmail.std" and then to install a new "xmail" program that looked something like this:

```
#!/bin/csh -f

setenv MM_NOTTTY 1
xmail.std $*
```

This should make xmail work with metamail, although, unlike most of the modified mailers described in this document, it will not ask the users for confirmation before running metamail applications.

1.4 Mailtool (SunTools Interface to Berkeley Mail)

Mailtool is a program that provides a graphical interface to Berkeley mail. It does most of its work by talking to the Berkeley mail program itself. Thus, if you want mailtool to work with metamail, you have to first of all upgrade your version of the Berkeley Mail program as described in the previous sections.

Once you have done this, everything would work fine automatically except that there’s no way for metamail to tell that it has been called non-interactively. You can tell it that this is the case (so that it won’t try to ask any questions) by setting the MM_NOTTTY environment variable. Thus, a simple way to fix mailtool would be to rename the "mailtool" program to be "mailtool.std" and then to install a new "mailtool" program that looked something like this:

```
#!/bin/csh -f

setenv MM_NOTTTY 1
mailtool.std $*
```

This should make mailtool work with metamail, although, unlike most of the modified mailers described in this document, it will not ask the users for confirmation before running metamail applications.

1.5 Imail (Bellcore MICE mailer)

All of the changes are localized to the file util.c In particular, there are two changes to be made:

1. In util.c, there is a routine called "listit" -- in my version it starts on line 82. The first line of that routine looks like this:

```
        debug("about to listit, %d lines\n",nlines);
```

Immediately following that line (i.e. between line 88 and line 89, in my version) **add the following code**:

```
#ifndef NOMETAMAIL
#include <sgtty.h>
        if (!getenv("NOMETAMAIL") && nontext(fp)) {
            struct sgttyb ttystatein, ttystateout;
            char Fname[100], Cmd[100], linebuf[1000];
            FILE *fp2;
            int chars = 0, code;

            sprintf(Fname, "/tmp/imail-metamail.%d.%d", getpid(), getuid());
            fp2 = fopen(Fname, "w");
            if (!fp2) {
                perror(Fname);
            } else {
                while( chars < p->len && fgets(linebuf,sizeof(linebuf),fp) != NULL) {
                    fputs(linebuf,fp2);
                    chars += strlen(linebuf);
                }
                fclose(fp2);
                sprintf(Cmd, "reset; metamail -z -m imail %s", Fname);
                gtty(fileno(stdin), &ttystatein);
                gtty(fileno(stdout), &ttystateout);
                code = system(Cmd);
                stty(fileno(stdin), &ttystatein);
                stty(fileno(stdout), &ttystateout);
                /* if (!code) return(0); */
                return(0); /* Don’t show raw datastream anyway! */
            }
        }
#endif
```

2. After the "listit" procedure, or at the very end of util.c, whichever you prefer, **add the following new routine**, which is used by the patch above:

```
#ifndef NOMETAMAIL
#include <ctype.h>

nontext(fp)
FILE *fp;
{
    char buf[1000], *s;
    int oldnl=nl, oldcnt=cnt, oldfptr, retval = 0;

    oldfptr = ftell(fp);
```

```
    while( cnt < nlines && fgets(buf,sizeof(buf),fp) != NULL) {
        if (buf[0] == '\n') break;
        for (s=buf; *s; ++s) if (isupper(*s)) *s = *s - 'A' + 'a';
       if (!strncmp(buf, "content-type:", 13)) {
            retval = 1;
            break;
       }
        nl++;
        cnt++;
    }
    nl = oldnl;
    cnt = oldcnt;
    fseek(fp, oldfptr, 0);
    return(retval);
}
#endif
```

These two changes should be all you need to do in order to make imail work with metamail, assuming that you already have the metamail binary installed somewhere on your search path.

1.6 PCS readmail/rdmail

All of the changes are localized to the file display.c In particular, there are two changes to be made:

1. In display.c, there is a routine called "display". The first line of that routine looks like this:

```
        messnum = messord[messnum];        /* convert to internal number */
```

Immediately following that line (i.e. between 13 and 14 in my version) **add the following code**:

```
#ifndef NOMETAMAIL
{
#include <sgtty.h>
   if (!getenv("NOMETAMAIL") && nontext(messnum)) {
       char Fname[100], Cmd[100], linebuf[1000];
       FILE *fp2;
       int code;
       struct sgttyb ttystatein, ttystateout;

       sprintf(Fname, "/tmp/readmail-metamail.%d.%d.%d", getpid(), getuid());
       fp2 = fopen(Fname, "w");
       if (!fp2) {
           perror(Fname);
       } else {
           fseek(curr.fp,messbeg[messnum],0);
           while (fgets (linebuf, sizeof(linebuf), curr.fp) != NULL && ftell(curr.fp) <=
messend[messnum]) {
               fputs(linebuf, fp2);
           }
           fclose(fp2);
           sprintf(Cmd, "metamail -m readmail -z %s %s", profile("page") ? "-p" : "",
Fname);
           gtty(fileno(stdin), &ttystatein);
           gtty(fileno(stdout), &ttystateout);
           code = system(Cmd);
           stty(fileno(stdin), &ttystatein);
           stty(fileno(stdout), &ttystateout);
           /* if (!code)) */ return(0);
       }
   }
}
#endif
```

2. At the very end of display.c, **add the following new routine**, which is used by the patch above:

```
#ifndef NOMETAMAIL
#include <ctype.h>

nontext(messnum)
int messnum;
{
    char buf[1000], *s;
```

```
    fseek(curr.fp,messbeg[messnum],0);
    while (fgets (buf, sizeof(buf), curr.fp) != NULL && ftell(curr.fp) < messend[messnum])
{
        if (buf[0] == '\n') return(0);
        for (s=buf; *s; ++s) if (isupper(*s)) *s = *s - 'A' + 'a';
        if (!strncmp(buf, "content-type:", 13)) return(1);
    }
    return(0);
}
#endif
```

These two changes should be all you need to do in order to make pcs readmail/rdmail work with metamail, assuming that you already have the metamail binary installed somewhere on your search path.

1.7 PCS sreadmail

All of the changes are localized to the file display.c In particular, there are two changes to be made:

1. In display.c, there is a routine called "display". The first line of that routine looks like this:

```
        messnum = messord[messnum];        /* convert to internal number */
```

Immediately following that line (i.e. between line 14 and line 15, in my version) **add the following code**:

```
#ifndef NOMETAMAIL
{
#include <sgtty.h>
   if (!getenv("NOMETAMAIL") && nontext(messnum)) {
       char Fname[100], Cmd[100], linebuf[1000];
       FILE *fp2;
       int code;
       struct sgttyb ttystatein, ttystateout;

       sprintf(Fname, "/tmp/sreadmail-metamail.%d.%d.%d", getpid(), getuid());
       fp2 = fopen(Fname, "w");
       if (!fp2) {
           perror(Fname);
       } else {
           fseek(curr.fp,messbeg[messnum],0);
           while (fgets (linebuf, sizeof(linebuf), curr.fp) != NULL && ftell(curr.fp) <=
messend[messnum]) {
               fputs(linebuf, fp2);
           }
           fclose(fp2);
           sprintf(Cmd, "reset ; metamail -m sreadmail -z -p %s", Fname);
           gtty(fileno(stdin), &ttystatein);
           gtty(fileno(stdout), &ttystateout);
           code = system(Cmd);
           stty(fileno(stdin), &ttystatein);
           stty(fileno(stdout), &ttystateout);
           clear();
           redisplay=TRUE;
           curr.pageno=0;
           helppage=0;
           /* if (!code)) */ return(0);
       }
   }
}
#endif
```

2. At the very end of display.c, **add the following new routine**, which is used by the patch above:

```
#ifndef NOMETAMAIL
#include <ctype.h>

nontext(messnum)
```

```
int messnum;
{
    char buf[1000], *s;

    fseek(curr.fp,messbeg[messnum],0);
    while (fgets (buf, sizeof(buf), curr.fp) != NULL && ftell(curr.fp) < messend[messnum])
{
        if (buf[0] == '\n') return(0);
        for (s=buf; *s; ++s) if (isupper(*s)) *s = *s - 'A' + 'a';
        if (!strncmp(buf, "content-type:", 13)) return(1);
    }
    return(0);
}
#endif
```

These two changes should be all you need to do in order to make pcs readmail/rdmail/sreadmail work with metamail, assuming that you already have the metamail binary installed somewhere on your search path.

1.8 MH -- Rand Message Handling System

All of the changes are localized to the file uip/show.c In particular, there are two changes to be made:

1. In uip/show.c, there is a label "go_to_it:". Shortly after that you'll find the following bit of code:

```
    if (nshow)
        proc = "/bin/cat";
    else {
        (void) putenv ("mhfolder", folder);
        if (strcmp (r1bindex (showproc, '/'), "mhl") == 0) {
            vec[0] = "mhl";
            (void) mhl (vecp, vec);
            done (0);
        }
        proc = showproc;
    }
```

Immediately preceding that code fragment (i.e. between between 245 and 246 in my version of uip/show.c) **add the following code**:

```
#ifndef NOMETAMAIL
    if (!getenv("NOMETAMAIL") && nontext(--msgnum, mp)) {
#include <sgtty.h>
        struct sgttyb ttystatein, ttystateout;
        char Cmd[120];
        FILE *fp;
        int code;

        sprintf(Cmd, "metamail -e -p -m MH %s/%d", mp->foldpath, msgnum);
        gtty(fileno(stdin), &ttystatein);
        gtty(fileno(stdout), &ttystateout);
        code = system(Cmd);
        stty(fileno(stdin), &ttystatein);
        stty(fileno(stdout), &ttystateout);
        exit(code);
    }
#endif
```

2. At the very end of uip/show.c, **add the following new routine**, which is used by the patch above:

```
#ifndef NOMETAMAIL
#include <ctype.h>

nontext(msgnum, mp)
int msgnum;
struct msgs *mp;
{
        FILE *fp;
        char line[1000], *s;

        sprintf(line, "%s/%d", mp->foldpath, msgnum);
        fp = fopen(line, "r");
```

```
        if (!fp) return(0);
        while (fgets(line, sizeof(line), fp)) {
            if (line[0] == '\n') {fclose(fp);return(0);}
            for (s=line; *s; ++s) if (isupper(*s)) *s = *s - 'A' + 'a';
            if (!strncmp(line, "content-type:", 13)) return(1);
        }
        fclose(fp);
        return(0);
}
#endif
```

These two changes should be all you need to do in order to make MH work with metamail, assuming that you already have the metamail binary installed somewhere on your search path.

1.9 XMH -- X11 Interface to Rand Message Handling System

All of the changes are localized to the file msg.c In particular, there are only two change to be made:

1. Near the beginning of the file (e.g. right after the last #include line), add the following code:

```
#ifndef NOMETAMAIL
struct mmdata {
    Msg msg;
    Scrn scrn;
};
void RedisplayMsg();

void NoCallMetamail (widget, client_data, call_data)
Widget  widget;
XtPointer *client_data;
XtPointer      call_data;
{
}

void CallMetamail (widget, mmd, call_data)
Widget  widget;
struct mmdata *mmd;
XtPointer      call_data;
{
    char TmpFileName[200];
    char Cmd[200];

    sprintf(TmpFileName, "/tmp/xmh.%d.%d", getpid(), getuid());
    sprintf(Cmd, "csh -c \"metamail -e -m XMH -x -d %s >& %s \"", MsgFileName(mmd->msg),
TmpFileName);
    fprintf(stderr, "Executing %s, please wait...\n", Cmd);
    system(Cmd);
    mmd->msg->source = CreateFileSource(mmd->scrn->viewwidget, TmpFileName, FALSE);
    RedisplayMsg(mmd->scrn);
    unlink(TmpFileName);
}
#endif
```

2. In msg.c, there is a procedure "SetScrnNewMsg". In that procedure you will find the following bit of code:

```
        msg->num_scrns++;
        msg->scrn = (Scrn *) XtRealloc((char *)msg->scrn,
                                  (unsigned) sizeof(Scrn)*msg->num_scrns);
        msg->scrn[msg->num_scrns - 1] = scrn;
        if ((msg->source == NULL) || (msg->toc == DraftsFolder))
            msg->source = CreateFileSource(scrn->viewwidget, MsgFileName(msg),
                                     scrn->kind == STcomp);
```

Immediately following that code fragment (i.e. between between 312 and 313 in my version of msg.c) **add the following code**:

```
#ifndef NOMETAMAIL
        if (!getenv("NOMETAMAIL")) {
#include <ctype.h>
            static XtCallbackRec yes_callbacks[] = {
                {CallMetamail, (XtPointer) NULL},
                {(XtCallbackProc) NULL,       (XtPointer) NULL}
            };

            static XtCallbackRec no_callbacks[] = {
                {NoCallMetamail, (XtPointer) NULL},
                {(XtCallbackProc) NULL,       (XtPointer) NULL}
            };
            FILE *fp;
            char line[1000], *s, *t, Query[200];
            static struct mmdata Mmdata;

            fp = fopen(MsgFileName(msg), "r");
            if (fp) {
                while (fgets(line, sizeof(line), fp)) {
                    if (line[0] == '\n') break;
                    for (s=line; *s; ++s) if (isupper(*s)) *s = *s - 'A' + 'a';
                    if (!strncmp(line, "content-type:", 13)) {
                        s = &line[13];
                        while (s && isspace(*s)) ++s;
                        t = index(s, ';');
                        if (t) {
                            *t = NULL;
                        } else {
                            t = index(s, '\n');
                            if (t) *t = NULL;
                        }
                        if (t--) {
                            while (isspace(*t) && (t > s)) {
                                *t-- = NULL;
                            }
                        }
                        if (strcmp(s, "text")) { /* not quite right */
                            Mmdata.msg = msg;
                            Mmdata.scrn = scrn;
                            yes_callbacks[0].closure = (XtPointer) &Mmdata;
                            no_callbacks[0].closure =  (XtPointer) NULL;
                            sprintf(Query, "This message is in %s format.\nDo you want to
try to run an external interpreter?", s);
                            if (getenv("MM_NOASK")) {
                                CallMetamail(scrn->viewwidget, &Mmdata, NULL);
                            } else {
                                PopupConfirm(scrn->tocwidget,      Query,      yes_callbacks,
no_callbacks);
                            }
                            break;
                        }
                    }
                }
                fclose(fp);
            }
        }
#endif
```

These changes should be all you need to do in order to make XMH work with metamail, assuming that you already have the metamail binary installed somewhere on your search path.

1.10 Rmail -- GNU Emacs mail reading package

Emacs being what it is, there are several versions of the rmail package floating around. Therefore the patch is particularly hard to describe, since your version of rmail may vary. The patch is therefore described in several steps.

1. **Make sure your rmail has an "rmail-show-message-hook" function.** If you look in the source file rmail.el, you will (probably) find a function called "rmail-show-message". This function itself varies significantly in different versions of rmail. Near the end of it, you might find a line of the form:

```
(run-hooks 'rmail-show-message-hook)
```

If there isn't such a line, you should add one -- probably right at the end of the function, before it returns.

2. You may now either make changes to the rmail source, or only to your own personal hook function.

2a. **If you want to change the rmail source**, put the following line immediately **before** the aforementioned "run-hooks" line:

```
(rmail-check-content-type)
```

2a. **If you want to change only your own customized rmail behavior**, put the following lines in your "~/.emacs" file:

```
(setq
 rmail-show-message-hook
 '(lambda()
        (rmail-check-content-type)))
```

(If you already had an rmail-show-message-hook function defined, you can just put the rmail-check-content-type call at the beginning of it.)

3. **Make sure you have the GNU "transparent.el" package installed.** This is not part of the standard distribution, but is widely available, and should be part of most FTP archives, etc.

4. **Put the following LISP code somewhere that emacs will find it.** In particular, if you modified rmail.el in step 2a, then it probably makes sense to put this code at the end of that file. If you just modified your own .emacs file, it probably makes sense to put this code in your .emacs file. The code you want is:

```
;;; Functions added for METAMAIL support

(require 'transparent)

(defvar rmail-never-execute-automatically t
        "*Prevent metamail from happening semi-automatically")

(define-key rmail-mode-map "!" 'rmail-execute-content-type)

(defun rmail-check-content-type ()
  "Check for certain Content Type headers in mail"
  (rmail-maybe-execute-content-type nil))

(defun rmail-execute-content-type ()
  "Check for certain Content Type headers in mail"
  (interactive)
  (rmail-maybe-execute-content-type t))

(defun rmail-handle-content-type (ctype override dotoggle)
  (let (oldpt
        (oldbuf (current-buffer))
        (fname (make-temp-name "/tmp/rmailct")))

    (cond
     ((and rmail-never-execute-automatically (not override))
      (progn
       (if dotoggle (rmail-toggle-header))
       (message (concat "You can use '!' to run an interpreter for this '"
                        ctype "' format mail."))))
     ((or override
          (getenv "MM_NOASK")
          (y-or-n-p (concat "Run an interpreter for this '"
                            ctype "' format mail? ")))
      (progn
       (save-restriction
        (goto-char (point-max))
        (setq oldpt (point))
        (goto-char 0)
        (widen)
        (write-region
         (point)
         oldpt
         fname
         'nil
         "silent"))
       (if dotoggle (rmail-toggle-header))
       (if
        (and window-system (getenv "DISPLAY"))
         (progn
          (switch-to-buffer-other-window "METAMAIL")
          (erase-buffer)
          (pop-to-buffer oldbuf)
          (start-process "metamail"  "METAMAIL" "metamail" "-m"
                         "rmail" "-p" "-x" "-d" "-z" "-q" fname)
          (message "Starting metamail.  Sending output to METAMAIL buffer."))
         (progn
          (switch-to-buffer "METAMAIL")
          (erase-buffer)
          (sit-for 0)
          (transparent-window
           "METAMAIL"
           "metamail"
```

```
           (list "-m" "rmail" "-p" "-d" "-z" "-q" "-R" fname)
           nil
          (concat
           "\n\r\n\r*****************************************"
           "*******************************\n\rPress any key "
           "to go back to EMACS\n\r\n\r***********************"
           "*************************************************\n\r")
          )))))
     (t (progn
          (if dotoggle (rmail-toggle-header))
          (message (concat "You can use the ’!’ keystroke to "
                          "execute the external viewing program.")))))))

(defun rmail-maybe-execute-content-type (dorun)
  "Check for certain Content Type headers in mail"
  (cond
   ((not (getenv "NOMETAMAIL"))
    (save-restriction
     (setq buffer-read-only ’nil)
     (let ((headend 0)
           (ctype "text")
           (old-min (point-min))
           (old-max (point-max))
           (needs-toggled nil)
           (opoint 0))
       (narrow-to-region (rmail-msgbeg rmail-current-message) (point-max))
       (goto-char (point-min))
       (forward-line 1)
       (if (= (following-char) ?1) (setq needs-toggled t))
       (goto-char (point-min))
       (search-forward "\n\n")
       (setq headend (point))
       (setq opoint (- headend 1))
       (goto-char (point-min))
       (setq case-fold-search ’T)
       (if needs-toggled (rmail-toggle-header))
       (cond
        ((search-forward "\ncontent-type:" opoint ’t)
         (progn
          (forward-word 1)    (backward-word 1)  ; Took care of white space
          (setq opoint (point))
          (re-search-forward "[;\n]" (point-max) t)
          (setq ctype (downcase (buffer-substring opoint (- (point) 1))))
          (goto-char (point-min)))))
       (cond ((not (string= ctype "text"))
              (rmail-handle-content-type
               ctype dorun needs-toggled))
             (needs-toggled
              (rmail-toggle-header))))))))
```

The above changes should be all you need to do in order to make rmail work with metamail, assuming that you already have the metamail binary installed somewhere on your search path.

1.11 VM -- Another GNU Emacs mail reading package

Emacs being what it is, there are several mail readers besides rmail floating around. The'yre all similar, however. The following patch for VM is extremely similar to the patch for rmail -- in fact, the contents of the hook functions are identical.

1. **Modify the definition of "vm-build-visible-header-alist"** to force the inclusion of the "Content-Type" and "Content-Transfer-Encoding" headers. In vm.el, you will find a definition that looks something like this:

```
(defun vm-build-visible-header-alist ()
  (let ((header-alist (cons nil nil))
  (vheaders vm-visible-headers)
   list)
    (setq list header-alist)
    (while vheaders
      (setcdr list (cons (cons (car vheaders) nil) nil))
      (setq list (cdr list) vheaders (cdr vheaders)))
    (setq vm-visible-header-alist (cdr header-alist))))
```

You should alter the declaration of "vheaders", so that it looks like this instead:

```
(defun vm-build-visible-header-alist ()
  (let ((header-alist (cons nil nil))
        (vheaders (append vm-visible-headers
                          '("Content-Type:" "Content-Transfer-Encoding:")))
        list)
    (setq list header-alist)
    (while vheaders
      (setcdr list (cons (cons (car vheaders) nil) nil))
      (setq list (cdr list) vheaders (cdr vheaders)))
    (setq vm-visible-header-alist (cdr header-alist))))
```

2. **Make sure your version of VM has a "vmail-show-message-hook" function.** It probably doesn't. However, if you look in the source file vm.el, you will (probably) find a function called "vm-show-current-message". Near the end of it, you might find a line of the form:

```
  (vm-update-summary-and-mode-line)
```

Immediately after that, you should add the line:

```
  (run-hooks 'vm-show-message-hook)
```

3. You may now either make changes to the vm source, or only to your own personal hook function.

3a. **If you want to change the vm source**, put the following line immediately **before** the aforementioned "run-hooks" line:

```
(vm-check-content-type)
```

3a. **If you want to change only your own customized vm behavior**, put the following lines in your "~/.emacs" file:

```
(setq
 vm-show-message-hook
 '(lambda()
        (vm-check-content-type)))
```

(If you already had an vm-show-message-hook function defined, you can just put the vm-check-content-type call at the beginning of it.)

4. **Make sure you have the GNU "transparent.el" package installed.** This is not part of the standard distribution, but is widely available, and should be part of most FTP archives, etc.

5. **Put the following LISP code somewhere that emacs will find it.** In particular, if you modified vm.el in step 2a, then it probably makes sense to put this code at the end of that file. If you just modified your own .emacs file, it probably makes sense to put this code in your .emacs file. The code you want is:

```
;;; Functions added for METAMAIL support

(require 'transparent)

(defvar vm-never-execute-automatically t "*Prevent metamail from happening semi-
automatically")

(define-key vm-mode-map "!" 'vm-execute-content-type)

(defun vm-check-content-type ()
  "Check for certain Content Type headers in mail"
  (vm-maybe-execute-content-type nil))

(defun vm-execute-content-type ()
  "Check for certain Content Type headers in mail"
  (interactive)
  (vm-maybe-execute-content-type t))

(defun vm-handle-content-type (ctype override)
  (let (oldpt
        (fname (make-temp-name "/tmp/rmailct")))

    (cond
     ((and vm-never-execute-automatically (not override))
      (progn
       (message (concat "You can use '!' to run an interpreter for this '"
                        ctype "' format mail."))))
     ((or override
          (getenv "MM_NOASK")
          (y-or-n-p (concat "Run an interpreter for this '"
                            ctype "' format mail? ")))
      (progn
       (save-restriction
        (goto-char (point-max))
```

```
        (setq oldpt (point))
        (goto-char 0)
        (widen)
        (write-region
         (point)
         oldpt
         fname
         'nil
         "silent"))
       (if
        (and window-system (getenv "DISPLAY"))
         (progn
          (switch-to-buffer-other-window "METAMAIL")
          (erase-buffer)
          (start-process "metamail"  "METAMAIL" "metamail" "-m"
                         "rmail" "-x" "-d" "-z" "-q" fname)
          (if vm-mail-buffer (pop-to-buffer vm-mail-buffer))
          (message "Starting metamail.  Sending output to METAMAIL buffer."))
         (progn
          (switch-to-buffer "METAMAIL")
          (erase-buffer)
          (sit-for 0)
          (transparent-window
           "METAMAIL"
           "metamail"
           (list "-m" "rmail" "-d" "-z" "-q" "-R" fname)
           nil
          (concat
           "\n\r\n\r*****************************************"
           "****************************\n\rPress any key "
           "to go back to EMACS\n\r\n\r***********************"
           "************************************************\n\r")
          )))))
     (t (progn
          (message (concat "You can use the '!' keystroke to "
                           "execute the external viewing program.")))))))

(defun vm-maybe-execute-content-type (dorun)
  "Check for certain Content Type headers in mail"
  (cond
   ((not (getenv "NOMETAMAIL"))
    (save-restriction
     (setq buffer-read-only 'nil)
     (let ((headend 0)
           (ctype "text")
           (old-min (point-min))
           (old-max (point-max))
           (opoint 0))
       (goto-char (point-min))
       (forward-line 1)
       (goto-char (point-min))
       (search-forward "\n\n")
       (setq headend (point))
       (setq opoint (- headend 1))
       (goto-char (point-min))
       (setq case-fold-search 'T)
       (cond
        ((search-forward "\ncontent-type:" opoint 't)
         (progn
          (forward-word 1)    (backward-word 1)  ; Took care of white space
          (setq opoint (point))
          (re-search-forward "[;\n]" (point-max) t)
```

```
          (setq ctype (downcase (buffer-substring opoint (- (point) 1))))
          (goto-char (point-min)))))
       (cond ((not (string= ctype "text"))
              (vm-handle-content-type
               ctype dorun))
             ))))))
```

6. **Watch out for an oddity in the terminal emulator package.** If you get the error message

```
Key sequence \277 uses invalid prefix characters
```

or something like that, this probably means that you have set your global variable "help-char" to a META key. The terminal emulator package doesn't like that at all, and you'll need to change it in order for the terminal emulator package, and these extensions to vm, to work properly.

The above changes should be all you need to do in order to make vm work with metamail, assuming that you already have the metamail binary installed somewhere on your search path.

1.12 CUI (Simplest Andrew Mail Reader)

NOTE: If you have the version of CUI that corresponds to Messages version 8.0 or later, you do not need this patch. If you have the version that corresponds to Messages 7.15 or later, you probably do not need this patch, either. If you have an earlier version, you need this patch.

In the file ams/msclients/cui/cui.c, there is a routine called "GetBodyFromCUID" The first line of that routine looks like this:

```
    debug(1,("GetBodyFromCUID %d\n", cuid));
```

Immediately after that line, which is between lines 1228 and 1229 in my version, add the following lines:

```
#ifndef NOMETAMAIL
    {
#include <sgtty.h>
        struct sgttyb ttystatein, ttystateout;
        char ctype[100], TmpFileName[1+MAXPATHLEN], Cmd[1+MAXPATHLEN];
        int ShouldDelete, code;
        extern int LinesOnTerminal;

        if (CUI_GetHeaderContents(cuid,(char *) NULL, HP_CONTENTTYPE, ctype, sizeof(ctype)
- 1) != NULL) {
            /* error already reported */
            return(-1);
        }
        if  (!getenv("NOMETAMAIL")  &&  ctype[0]  &&  strnicmp(ctype,  "x-be2",  5)  &&
strncmp(ctype, "text")) {
            if (CUI_GetBodyToLocalFile(cuid, TmpFileName, &ShouldDelete)) {
                return(-1); /* error reported */
            }
            sprintf(Cmd, "metamail -m cui %s %s %s", (LinesOnTerminal > 0) ? "-p" : "",
ShouldDelete ? "-z" : "", TmpFileName);
            gtty(fileno(stdin), &ttystatein);
            gtty(fileno(stdout), &ttystateout);
            code = system(Cmd);
            stty(fileno(stdin), &ttystatein);
            stty(fileno(stdout), &ttystateout);
            /* if (code) */ return(0);
        }
    }
#endif
```

This simple change should be all you need to do in order to make CUI work with metamail, assuming that you already have the "metamail" executable installed somewhere on your search path.

1.13 VUI (Terminal-oriented Andrew Mail Reader)

NOTE: If you have the version of VUI that corresponds to Messages version 8.0 or later, you do not need this patch. If you have the version that corresponds to Messages 7.15 or later, you probably do not need this patch, either. If you have an earlier version, you need this patch.

In the file ams/msclients/vui/vuipnl.c, there is a routine called "InitBodyData" That routine starts out with the following line of code:

```
    char filename[MAXPATHLEN+1];
```

Immediately after that code, which is between lines 2386 and 2387 in my version, add the following lines:

```
#ifndef NOMETAMAIL
#include <sgtty.h>
        struct sgttyb ttystatein, ttystateout;
        char ctype[100], TmpFileName[1+MAXPATHLEN], Cmd[1+MAXPATHLEN];
        int ShouldDelete, cuid = CuidFromMsgno(msgno), code;

        if (CUI_GetHeaderContents(cuid,(char *) NULL, HP_CONTENTTYPE, ctype, sizeof(ctype)
- 1) != NULL) {
            /* error already reported */
            return(-1);
        }
        if  (!getenv("NOMETAMAIL")  &&  ctype[0]  &&  ULstrncmp(ctype,  "x-be2",  5)  &&
ULstrcmp(ctype, "text")) {
            if (CUI_GetBodyToLocalFile(cuid, TmpFileName, &ShouldDelete)) {
                return(-1); /* error reported */
            }
            sprintf(Cmd, "reset; metamail -m vui -p %s %s", ShouldDelete ? "-z" : "",
TmpFileName);
            gtty(fileno(stdin), &ttystatein);
            gtty(fileno(stdout), &ttystateout);
            code = system(Cmd);
            stty(fileno(stdin), &ttystatein);
            stty(fileno(stdout), &ttystateout);
            RedrawScreen(0);
            /* if (code) */ return(MENU_EMPTY);
        }
#endif
```

This simple change should be all you need to do in order to make VUI work with metamail, assuming that you already have the "metamail" executable installed somewhere on your search path.

1.14 Messages (Andrew Multimedia Mail Reader)

NOTE: If you have Messages version 8.0 or later, you do not need this patch. If you have Messages 7.15 or later, you probably do not need this patch, either. If you have an earlier version, you need this patch.

There are two changes, both localized to the file atkams/messages/lib/text822.c:

1. Near the beginning of atkams/messages/lib/text822.c, there are the following #include lines:

```
#include <stylesht.ih>
#include <environ.ih>
#include <fnote.ih>
#include <ams.h>
```

Immediately after those lines, insert the following lines:

```
#ifndef NOMETAMAIL
#include <sys/param.h>
#include <fdphack.h>
#undef popen /* BOGUS -- should be handled by fdphack */
#undef pclose /* ditto */
#include <ams.ih>
#include <message.ih>
#include <im.ih>

MetaOutput(fp, self)
FILE *fp;
struct text *self;
{
    char buf[1000];

    if (fgets(buf, sizeof(buf), fp) != NULL) {
        text_AlwaysInsertCharacters(self, text_GetLength(self), buf, strlen(buf));
        text_NotifyObservers(self, 0);
        return;
    }
    if (errno != EWOULDBLOCK) {
        im_RemoveFileHandler(fp);
        pclose(fp);
        strcpy(buf, "\n--- Command execution terminated ---\n");
        text_AlwaysInsertCharacters(self, text_GetLength(self), buf, strlen(buf));
    }
    text_NotifyObservers(self, 0);
}
#endif
```

2. In atkams/messages/lib/text822.c, there is a routine called "text822__ReadIntoText" Near the end of that very long routine, you will find some code that looks like this:

```
        } else if (!amsutil_lc2strncmp("troff", sfmttype, strlen(sfmttype))) {
            char **resources = (char **) amsutil_BreakDownResourcesIntoArray(fmtresources);

            rofftext_ReadRoffIntoText(d, fp, ShowPos, resources);
```

```
            if (resources) free(resources);
            ReadRaw = FALSE;
        }
```

The patch needs to be added AFTER the line that says "ReadRaw = FALSE" but before the next line, which is just a closing brace. (In my version, this means the patch goes between lines 353 and 354.) The patch to insert is as follows:

```
#ifndef NOMETAMAIL
        } else if (!environ_Get("NOMETAMAIL") && IsReallyTextObject && amsutil_lc2strncmp("text", sfmttype, strlen(sfmttype))) {
            /* IsReallyTextObject test ensures we don’t run metamail when printing!!! */
                char TmpFileName[1+MAXPATHLEN], LineBuf[1000], Cmd[1+MAXPATHLEN], Msg[50+MAXPATHLEN], TmpFile2[1+MAXPATHLEN];
            FILE *fp2;

            sprintf(Msg, "Do you want to run an interpreter for this ’%s’ format mail", sfmttype);
            if (environ_Get("MM_NOASK")
                || ams_GetBooleanFromUser(ams_GetAMS(), Msg, TRUE)) {
                ams_CUI_GenLocalTmpFileName(ams_GetAMS(), TmpFileName);
                ams_CUI_GenLocalTmpFileName(ams_GetAMS(), TmpFile2);
                fp2 = (FILE *) fopen (TmpFileName, "w");
                if (fp2) {
                    fseek(fp, 0, 0);
                    while (fgets(LineBuf, sizeof(LineBuf), fp)) {
                        fputs(LineBuf, fp2);
                    }
                    fclose(fp2);
                    sprintf(Cmd, "metamail -m messages -z -x -d -q %s 2>&1", TmpFileName);
                    sprintf(Msg, "Executing: %s\n", Cmd);
                    linelen = strlen(Msg);
                    text822_AlwaysInsertCharacters(d, ShowPos, Msg, linelen);
                    ShowPos += strlen(Msg);
                    fp2 = (FILE *) popen(Cmd, "r");
                    im_AddFileHandler(fp2, MetaOutput, d, 0);
                }
            }
#endif
```

This simple change should be all you need to do in order to make Messages work with metamail, assuming that you already have the "metamail" executable installed somewhere on your search path.

1.15 Elm (Mail system from HP)

All of the changes are localized to the file src/showmsg.c In particular, there are two changes to be made:

1. In src/showmsg.c, there is a routine called "show_msg" my version it starts on line 49. About a page down in that routine, you'll find something that looks like this:

```
        memory_lock = FALSE;

        /* some explanation for that last one - We COULD use memory locking
           to speed up the paging, but the action of "ClearScreen" on a screen
           with memory lock turned on seems to vary considerably (amazingly so)
           so it's safer to only allow memory lock to be a viable bit of
           trickery when dumping text to the screen in scroll mode.
           Philosophical arguments should be forwarded to Bruce at the
           University of Walamazoo, Australia, via ACSNet  *wry chuckle* */
```

Immediately following that fragment (i.e. between line 114 and line 115, in my version) **add the following code**:

```
#ifndef NOMETAMAIL
        if (!getenv("NOMETAMAIL") && nontext(current_header)) {
#include <sgtty.h>
            char fname[100], Cmd[200], line[VERY_LONG_STRING];
            struct sgttyb ttystatein, ttystateout;
            int code;
            long lines = current_header->lines;
            FILE *fpout;

            if (fseek(mailfile, current_header->offset, 0) != -1) {
                sprintf(fname, "/tmp/elm-metamail.%d.%d", getpid(), getuid());
                fpout = fopen(fname, "w");
                if (fpout) {
                    while (lines > 0L) {
                        fgets(line, VERY_LONG_STRING, mailfile);
                        fputs(line, fpout);
                        --lines;
                    }
                    fclose(fpout);
                    sprintf(Cmd, "reset; metamail -p -z -m Elm %s", fname);
                        Raw(OFF);
                    code = system(Cmd);
                        Raw(ON);
                        PutLine0(LINES,0,"Press any key to return to index.");
                        (void) ReadCh();
                    /* if (code) */ return(0);
                }
            }
        }
#endif
```

2. At the very end of src/showmsg.c, **add the following new routine**, which is used by the patch above:

```
#ifndef NOMETAMAIL
nontext(ch)
struct header_rec *ch;
{
    long lines = ch->lines;
    char line[VERY_LONG_STRING], *s, *t;

    if (fseek(mailfile, ch->offset, 0) == -1) return(-1);
    while (lines > 0L) {
        fgets(line, VERY_LONG_STRING, mailfile);
        --lines;
        if (line[0] == '\n') return(0);
        for (s=line; *s; ++s) if (isupper(*s)) *s = *s - 'A' + 'a';
        if (!strncmp(line, "content-type:", 13)) {
            s = &line[13];
            while (s && isspace(*s)) ++s;
            t = index(s, ';');
            if (!t) t = index(s, '\n');
            if (t) *t-- = NULL;
            while (t && *t && t > s && isspace(*t)) *t-- = NULL;
            if (strcmp(s, "text")) return(1);
        }
    }
    return(0);
}
#endif
```

These two changes should be all you need to do in order to make Elm work with metamail, assuming that you already have the "metamail" executable installed somewhere on your search path.

1.16 Msgs (Berkeley Bulletin Board system)

All of the changes are localized to the file msgs.c In particular, there are four changes to be made:

1. In msgs.c, there are lots of global declarations near the top. In my version, the last of them (around line 114) looks like this:

```
jmp_buf tstpbuf;
```

Immediately after that line, **add the following code**:

```
#ifndef NOMETAMAIL
int NonTextMessage;
#endif
```

2. Further down in msgs.c, there is a routine named "prmesg" (around line 580) which starts with the following declarations:

```
FILE *outf, *inf;
int c;
```

Immediately after these declaration, **add the following code**:

```
#ifndef NOMETAMAIL
        if (!getenv("NOMETAMAIL") && NonTextMessage) {
            char Fname[100], Cmd[120];
            FILE *fp;
            int code;
            struct sgttyb ttystatein, ttystateout;

            sprintf(Cmd, "metamail %s -m Mail %s", pause ? "-p" : "", fname);
            gtty(fileno(stdin), &ttystatein);
            gtty(fileno(stdout), &ttystateout);
            code = system(Cmd);
            stty(fileno(stdin), &ttystatein);
            stty(fileno(stdout), &ttystateout);
            return;
          }
#endif
```

3. Still further down in msgs.c, there is a routine named "gfrsub" (around line 775). A page or so down in that routine, there is code that looks like this:

```
        while (fgets(inbuf, sizeof inbuf, infile)
            && !(blankline = (inbuf[0] == '\n'))) {
                /*
                 * extract Subject line
                 */
                if (!seensubj && strncmp(inbuf, "Subj", 4)==0) {
                        seensubj = YES;
                        frompos = ftell(infile);
                        strncpy(subj, nxtfld(inbuf), sizeof subj);
                }
```

Immediately before this code fragment, add the following three lines:

```
#ifndef NOMETAMAIL
	NonTextMessage = 0;
#endif
```

Immediately after this code fragment (around line 840 in my version) add the following code:

```
#ifndef NOMETAMAIL
         {
           char *s, *t;
           for (s=inbuf; *s; ++s) if (isupper(*s)) *s = *s - 'A' + 'a';
           if (!strncmp(inbuf, "content-type:", 13)) {
             s = inbuf + 13;
             while (s && isspace(*s)) ++s;
             t = (char *) index(s, ';');
             if (t) *t = NULL; else t = s+strlen(s);
             while (--t > s && isspace(*t)) *t = NULL;
             NonTextMessage = strcmp(s, "text");
           }
         }
#endif
```

These three changes should be all you need to do in order to make msgs work with metamail, assuming that you already have the "metamail" executable installed somewhere on your search path.