A Multi-Media E-Mail Tutorial With MH

Jerry Sweet <jsweet@irvine.com>

ABSTRACT

This tutorial document introduces multi-media e-mail facilities to MH users. In particular, this tutorial discusses how to read and to send multi-media e-mail messages using MH. The goals are to provide enough information to prepare readers of this tutorial to customize their multi-media e-mail environments, and to encourage readers to tackle the manual page for `mhn`, the MH program that handles multi-media messages.

$Id: mhn-tutorial.tex,v 0.4 93/03/17 02:05:56 jsweet Rel $

Contents

1 Scope **3**

2 Introduction **3**

3 Simple Multi-Media Message Composition **5**

4 Multi-Media E-Mail Examples **8**

4.1 Weather Maps 8

4.2 A Periodical 14

5 Setting Up For MIME **17**

5.1 A Simple Setup 18

5.2 Setting Up For An X Window System Display 21

5.3 Examples of `mhn_defaults` 25

6 MIME and MH **26**

7 Mhn Explained **27**

7.1 Profile Entries 29

7.2 Displaying Messages 30

7.3 Composing Messages 32

7.4 Storing Message Parts 35

7.5 Other Facilities 36

8 More About External Body Parts **37**

A General Information about MH **41**

B Pre-Defined MIME Types **42**

C Richtext **44**

D Audio Setup **48**

D.1 SPARCstation 49

D.2 HP 9000/710 52

E On-Line Resources **55**

F Acknowledgements **62**

G References **63**

1 Scope

This document assumes that you have knowledge of UNIX and of MH, although not necessarily of MIME (Multipurpose Internet Mail Extensions), the Internet multi-media e-mail specifications.

A few simple ways to use MH to compose and to read multi-media e-mail are presented. It's assumed that you have a pre-configured MH mail system with reasonable defaults.

Skip to the end of this document for some references to books and to papers supplied with the MH software distribution that may be of some help to you if you're unfamiliar with MH.

2 Introduction

Most of MH's multi-media-related functions are implemented in one program, `mhn`. `Mhn` performs four functions: message composition, message display, message scanning, and message decomposition.

Using `mhn`, you can create and read e-mail messages containing these things:

- images
- sounds
- tar files
- PostScript
- pointers to FTPable[1] files
- other stuff

Certain hardware and software resources are needed in conjunction with `mhn` to make images and sounds from multi-media messages available to you. An X terminal makes it possible to view certain standard image types. A

[1]FTP: *v.,* to transfer a file from a remote Internet host; *n.,* 1. the Internet file transfer protocol; 2. the program by which a user performs file transfers via the file transfer protocol; –able, *adj.,* capable of being transferrable via FTP.

workstation with audio capabilities makes it possible to hear certain audio formats. (Unfortunately, as of this writing, the X Window System specification doesn't accommodate audio data, and only certain types of workstations will play sounds for you.)

If you don't have these resources, well...you're probably not going to have quite as much fun, but you can still use some of the most useful multi-media features of MH.

You'll also need correspondents who can send and receive multi-media e-mail in MIME format. Anyone using MH 6.8 will do; even if you're just corresponding with yourself for test purposes.

MH version 6.8 was the first major release of MH to contain the MIME extensions. To find out whether your version of MH supports MIME, try running this command:

```
mhn -help
```

If you see the `mhn` help message, then you've got multi-media MH. The help message should look something like this:

```
syntax: mhn [+folder] [msgs] [switches]
  switches are:
  -[no]auto
  -[no]ebcdicsafe
  -(form) formfile
  -[no]headers
  -[no]list
  -part number
  -[no]realsize
  -[no]rfc934mode
  -[no]serialonly
  -[no]show
  -[no]store
  -type content
  -[no]verbose
  -(help)
```

If the `mhn` command isn't available, then you'll see something like this:

```
mhn: Command not found.
```

In this tutorial, we're going to dive right into some examples, and save the more detailed explanations for later.

3 Simple Multi-Media Message Composition

Here's an example of how you can compose a multi-media e-mail message. The prompts and other MH program outputs are shown in `typewriter` style; your responses are shown in *slanted* style.

```
% comp
To:   ghb@white-house.gov
Cc:
Subject:   you're sacked!
-------
```

As you know, your contract of employment with us contained an "at will" clause. We have chosen to exercise that clause. A security guard shall escort you to your desk, from which you may obtain your personal effects, and then escort you from the premises. A severance check shall be mailed to you shortly. Good day to you, sir.

#audio/basic /goodies/sounds/rude-noise.au

#image/gif /goodies/images/rude-gesture.gif

^D

```
What now? edit mhn -list
```

```
msg part  type/subtype               size description
  1        multipart/mixed            45K
    1      text/plain                 332
    2      audio/basic               8961
    3      image/gif                  36K
```

```
What now? send
```

As you can see from the output of the `edit mhn` command, this message has three parts:

1. a plain text part (the introductory paragraph);
2. an audio part inserted via the line beginning with `#audio/basic`;
3. an image part inserted via the line beginning with `#image/gif`.

In the interest of good taste, we shan't show you the image, nor play for you the audio. The names of the files should be self-explanatory.

The lines beginning with hash marks (`#`) in the message above are examples of `mhn`'s special directives.

Each "`#`" directive indicates a message part, and identifies what type of information is to be carried in that part. For example, a message part containing audio data might have either type `audio/basic` or type `audio/x-next`. (These are both `audio` types; the *subtypes* are `basic` and `x-next`.) A list of the pre-defined types and subtypes is given in appendix B.

By using variants of the directives shown above, it's possible to tell `mhn` to collect into a message the audio or image data produced by running a program.

At the `What Now?` prompt, when you typed the command `edit mhn`, a new draft was created, replacing your previous draft. Thereafter, if instead of typing `send`, you had typed `edit emacs`, you would see MIME constructs in place of your original `mhn` directives, looking something like this:

```
To: ghb@white-house.gov
Subject: you're sacked!
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="----- =_aaaaaaaaaa0"

------- =_aaaaaaaaaa0
Content-Type: text/plain; charset="us-ascii"

As you know, your contract of employment with us contained
... [remainder of introductory paragraph omitted]

------- =_aaaaaaaaaa0
Content-Type: audio/basic
Content-Transfer-Encoding: base64

LnNuZAAAACgAAAk/AAAAQAAH0AAAAABdGVycmllciBiYXJrAAAAAFdSUVVdcuXc3OLtdmFbWV5r
897Z2d/tfGliYWVu8eHXz8vL0ONdS0ZGTVhq49PMy9X4W05NT1VfftnKyMvQ3vlhTktbeFhIQ0t8
... [ many lines of encoded audio data omitted ]

------- =_aaaaaaaaaa0
Content-Type: image/gif
Content-Transfer-Encoding: base64

R0lGODdhgALgAcMAAAAAAIAAAACAAICAAAAAgIAAgACAgICAgMDAwP8AAAD/AP//AAAA//8A/wD/
... [ many lines of encoded image data omitted ]

------- =_aaaaaaaaaa0--
```

Looks horrendous, doesn't it? You may be relieved to know that this tutorial doesn't explain all the gory details of the MIME format per se, and you don't really have to know them to use MH's multi-media features, because `mhn` shields you. If you're interested, there are references to documents that cover MIME at the end of the tutorial.

If you want to make corrections to a draft already processed by `edit mhn`, you can recover the original draft, containing your original `mhn` directives, from a file that `mhn` names as *,msg*.orig, where *msg* is whatever the designation of the original draft was. Example: `~/Mail/drafts/,1.orig`.

Using `mhn`'s directives, you don't need a bit-mapped display or a digital audio device to compose a multi-media e-mail message. But you do need a graphical display to view image parts and you do need a speaker to listen

to audio parts. There are other types of useful multi-media components that can be displayed and manipulated with nothing more than a plain old fixed-font 24×80 ASCII terminal.

4 Multi-Media E-Mail Examples

This section gives examples of how more complicated kinds of multi-media e-mail messages can be created.

Your system administrator has probably set up some reasonable `mhn` defaults, but if you run into problems when trying to display the results of these examples, you may need to do some additional setup work for yourself. Section 5 discusses setting up your profile for `mhn`.

4.1 Weather Maps

Here's a fun thing I did with `mhn`: I wrote a program, `mail.weather.ext`, which mails out a weather map image of the continental United States, a key to the weather symbols, and a corresponding satellite image.

The GIF files[2] containing these images are obtained freely from an Internet site.

I view these messages on a Sun IPC workstation with a 17-inch color display. Other persons in offices next to mine can view the images on HP 9000/700 series workstations with 19-inch color displays or on their monochrome X terminals—although the maps don't look quite as nice in monochrome.

By using the UNIX `cron` facility, I've arranged for the maps to be mailed out every weekday morning at 10:30. This is the line in my `crontab` that does it:

```
30 10 * * 1-5 /home/jsweet/bin/mail.weather.ext weather-fiends \
```

[2]GIF is a popular interchange format for graphical data. GIF is also a service mark of CompuServe, Inc.

```
>/dev/null 2>&1
```

Below is an example of an `mhn` draft template produced automatically by `mail.weather.ext`. The template contains two major pieces: an explanation, rendered in *richtext* format (we'll get to that by and by), of the images in the message, and a collection of *external body parts* that refer to GIF-format image files residing on a remote host.

```
To: weather-fiends
Subject: weather report: 02/25 0900 GMT
--------
#<text/richtext [explanation]
<paragraph>You may FTP three images:</paragraph>
<paragraph><indent><fixed>
SA022509.GIF: surface analysis at 0900 GMT 02/25<nl>
WXKEY.GIF:    map symbol key<nl>
CI022508.GIF: satellite photo at 0800 GMT 02/25
</fixed></indent></paragraph>
#
#begin [weather report: 02/25 0900 GMT] parallel
#@image/gif [surface analysis + radar summary] \
        access-type=anon-ftp; \
        name="SA022509.GIF"; \
        directory="wx"; \
        site="vmd.cso.uiuc.edu"; \
        expiration="26 Feb 93 09:00:00 GMT"; \
        size=50000
#@image/gif [weather map symbol key] \
        access-type=anon-ftp; \
        name="WXKEY.GIF"; \
        directory="wx"; \
        site="vmd.cso.uiuc.edu"; \
        size=18867
#@image/gif [satellite photo] \
        access-type=anon-ftp; \
        name="CI022508.GIF"; \
        directory="wx"; \
        site="vmd.cso.uiuc.edu"; \
        expiration="26 Feb 93 09:00:00 GMT"; \
        size=170000
#end
```

As you may have noted from the `expiration` attributes in the example above, these particular weather image files are long gone.

The structure of the message appears as follows, as shown using the command `mhn -list`, with some blank lines and indentation added for slightly easier reading:

```
msg part  type/subtype              size description
 1        multipart/mixed           1541

    1     text/richtext              288 explanation

    2     multipart/parallel         939 weather report: 02/25 0900 GMT

      2.1  message/external-body      78
           image/gif                    surface analysis + radar summary

      2.2  message/external-body      68
           image/gif                    weather map symbol key

      2.3  message/external-body      61
           image/gif                    satellite photo
```

When this message is shown via the MH command `show`, message part 1, a *richtext* part giving an explanation of the contents of the message, is displayed.

Then a series of prompts is provided for retrieving the *external bodies* named in part 2. The retrieval is accomplished semi-automatically, using anonymous FTP, from Internet host `vmd.cso.uiu.edu`, where the weather images are stored.

```
Date:    Thu, 25 Feb 1993 01:41:13 PST
To:      jsweet@irvine.com

From:    Jerry Sweet <jsweet@irvine.com>
Subject: weather report: 02/25 0900 GMT

Mime-Version: 1.0
```

```
You may FTP three images:


        SA022509.GIF: surface analysis at 0900 GMT 02/25
        WXKEY.GIF:    map symbol key
        CI022508.GIF: satellite photo at 0800 GMT 02/25


Press RETURN to continue (or 'q' to quit):

Retrieve SA022509.GIF (content 2.1)
    using anonymous FTP from site vmd.cso.uiuc.edu (50000 octets)? y
Make cached, publicly-accessible copy? y

Retrieve WXKEY.GIF (content 2.2)
    using anonymous FTP from site vmd.cso.uiuc.edu (18867 octets)? y
Make cached, publicly-accessible copy? y

Retrieve CI022508.GIF (content 2.3)
    using anonymous FTP from site vmd.cso.uiuc.edu (170000 octets)? y
Make cached, publicly-accessible copy? y
```

As the different external body parts are retrieved, the images pop up on the display (assuming that you have a graphical display). In my configuration, the `xv` application is given the task of showing the retrieved images on my X Window System display.

The prompts regarding "cached, publicly-accessible" copies have to do with the caching feature of `mhn` which may or may not be enabled at your site. You can enable caching for yourself with an appropriate `mhn` profile entry, discussed in section 5. If someone else at your site has already read the same message, and has made cached, publicly-accessible copies of the images, then you are likely to see prompts like this one instead:

```
Use cached copy /r/mhn-cache/<24860.731615403.1@fester.irvine.com>
        of size 25551 octets (content 2.1)?
```

In this example, the cache directory is `/r/mhn-cache/`. The horrible-looking file name surrounded by <angle brackets> corresponds to the unique *Content-ID* of the external body part. The most useful bit of information is the

part number, `2.1` in this example. From the previous listing of the message structure, you can see that part `2.1` refers to the "surface analysis + radar summary."

If caching isn't enabled, then the relevant prompts don't appear; the external body parts have to be retrieved from the remote host every time you want to see them, and the external body parts are deleted as soon as you're done with viewing them.

The *richtext* part of the weather draft message above was this:

```
#<text/richtext [explanation]
<paragraph>You may FTP three images:</paragraph>
<paragraph><indent><fixed>
SA022509.GIF: surface analysis at 0900 GMT 02/25<nl>
WXKEY.GIF:    map symbol key<nl>
CI022508.GIF: satellite photo at 0800 GMT 02/25
</fixed></indent></paragraph>
#
```

As you may have guessed, the *richtext* body part consists of text with some simple format commands, such as `<fixed>` and `<indent>`. You can see for yourself above what effects the richtext formatting commands have.

The richtext area of the message is delimited at the beginning by the `mhn` directive `#<text/richtext` and at the end by `#` alone on a line. In contrast, a text area of a message draft not specifically delimited by `mhn` directives is assumed to be of type `text/plain`.

The *description* of the richtext part is given in square brackets: `[explanation]`. The description shows up when the contents of a MIME message are listed via the `mhn -list` option.

A summary of richtext is given in appendix C.

The weather map image retrieval information begins with this directive:

```
#begin [weather report: 02/25 0900 GMT] parallel
```

The `#begin parallel` directive (ignoring the description in square brackets shown above) allows all three images that follow to be displayed at the same time. Without the `#begin parallel` directive, each image would be shown one at a time, and you would have to quit the image display application (`xv` in my case) each time in order to see the next image.

The scope of the `#begin` directive is ended, naturally enough, with an `#end` directive.

As you probably guessed by reading the retrieval prompts above, the weather map images themselves are not included in the message. Instead, the message contains references to files available via anonymous FTP on a remote host, in this case, `vmd.cso.uiuc.edu`.

One of the external file references is this:

```
#@image/gif [surface analysis + radar summary] \
        access-type=anon-ftp; \
        name="SA022509.GIF"; \
        directory="wx"; \
        site="vmd.cso.uiuc.edu"; \
        expiration="26 Feb 93 09:00:00 GMT"; \
        size=50000
```

This directive contains these pieces:

1. The external part indicator, the "`@`" character.
2. The *type*, `image/gif`, specifying a part containing GIF format data.
3. The *description*, given in square brackets, describing the content as "surface analysis + radar summary."
4. A set of *attribute*`=`*value* pairs:

   `access-type=anon-ftp` specifies that the file is to be retrieved via anonymous FTP.

   `name="SA022509.GIF"` specifies the name of the file on the remote Internet host. The name happens to refer to the month (02), day

(25) and hour (09, Greenwich Mean Time) when the weather data were assembled to create the image.

`directory="wx"` specifies the name of the directory in which the file resides.

`site="vmd.cso.uiuc.edu"` specifies the name of the remote Internet host. This particular host resides with the Department of Computer Science at the University of Illinois, Urbana-Champaign.

`expiration="26 Feb 93 09:00:00 GMT"` an optional attribute, specifies the date and time after which the file isn't expected to be available. At present, MH ignores this field, but other MIME-capable user agents might not.

`size=50000` also an optional attribute, is an estimate of the file size in bytes, provided for informational purposes only.

More is said about external body parts in section 8.

4.2 A Periodical

Dr. Marshall T. Rose does a useful thing with multi-media e-mail. He mails out each issue of his SNMP-related newsletter, *The Simple Times*, in these formats:

1. in a PostScript form, with beautiful typesetting and a two-column page layout, suitable for printing on a laser printer;
2. in *richtext* form (to be explained later on), suitable for display on a mildly intelligent ASCII terminal;
3. in a plain text form.

When you subscribe to *The Simple Times*, you specify in which form you want to receive the newsletter.

(SNMP is the Simple Network Management Protocol, having naught to do with MIME or MH per se.)

The process by which Dr. Rose creates an issue of *The Simple Times* begins with documents in LaTeX markup language format. A filter program is applied to strip the LaTeX constructs to produce plain ASCII text files. Another filter program is applied to convert the LaTeX constructs into richtext constructs. (The plain text and richtext files may require some additional editing to look reasonable.) The PostScript version is the result of applying LaTeX to the original version of the document followed by an appropriate PostScript-producing post-processor, such as `dvips`.

Dr. Rose was kind enough to provide this `mhn` draft template as an example of how he created the plain text MIME version of Volume 1, Number 4 of *The Simple Times*:

```
Reply-to: The Simple Times <st-editorial@simple-times.org>
To:       The Simple Times Subscribers <st-editorial@simple-times.org>
Dcc:      st-mime-subscribers
Fcc:      outbox, simple-times/1.4, /var/lists/simple-times/mime
Subject:  The Simple Times, volume 1, number 4

#begin      [The Simple Times]
#begin      [Issue Information]
Content-Description: Masthead

The Simple Times(tm)

------------------------------------------------------------------------
The Bi-Monthly Newsletter of SNMP Technology, Comment, and Events   (sm)
Volume 1, Number 4                                 September/October, 1992
------------------------------------------------------------------------
#text/plain [READ-ME]                   ../../lib/mime/read-me.txt
#text/plain [Disclaimer]                ../../lib/mime/disclaimer.txt
#end
#begin      [Issue Contents]
#text/plain [Technical Article]         technical.txt
#text/plain [Industry Comment]          comment.txt
#begin      [Featured Columns]
#text/plain [Applications and Directions]applications.txt
#text/plain [Ask Dr. SNMP]              faq.txt
#text/plain [Security and Protocols]    security.txt
#text/plain [Standards]                 standards.txt
#text/plain [Working Group Synopses]    synopses.txt
#end
```

```
#begin      [Miscellany]
#text/plain [Announcements]             announcements.txt
#text/plain [Recent Publications]       publications.txt
#text/plain [Activities Calendar]       calendar.txt
#end
#end
#begin      [Administrative Information]
#text/plain [Publication Information]   ../../lib/mime/publication.txt
#text/plain [Submissions]               ../../lib/mime/submissions.txt
#text/plain [Subscriptions Information] ../../lib/mime/subscriptions.txt
#end
#end
```

The structure of the message appears as follows, as shown using `mhn -list`, with some blank lines and indentation added for slightly easier reading:

```
msg part  type/subtype          size description

  1        multipart/mixed       93K The Simple Times

    1      multipart/mixed      1682 Issue Information
      1.1   text/plain           314 Masthead
      1.2   text/plain           377 READ-ME
      1.3   text/plain           668 Disclaimer

    2      multipart/mixed       88K Issue Contents
      2.1   text/plain           16K Technical Article
      2.2   text/plain          7864 Industry Comment

      2.3   multipart/mixed      60K Featured Columns
        2.3.1 text/plain        7223 Applications and Directions
        2.3.2 text/plain        7568 Ask Dr. SNMP
        2.3.3 text/plain        7820 Security and Protocols
        2.3.4 text/plain         14K Standards
        2.3.5 text/plain         22K Working Group Synopses

      2.4   multipart/mixed     3172 Miscellany
        2.4.1 text/plain        2345 Announcements
        2.4.2 text/plain         118 Recent Publications
        2.4.3 text/plain         361 Activities Calendar

    3      multipart/mixed      3077 Administrative Information
```

```
    3.1   text/plain          1032 Publication Information
    3.2   text/plain          1155 Submissions
    3.3   text/plain           534 Subscriptions Information
```

The richtext version has the same structure as the plain text version, except that richtext-format files are used in place of the plain text files.

Because the PostScript version is one large file, suitable for printing, the PostScript message doesn't use the full structure shown above for the plain text version; just a single part of type `application/postscript` is specified:

```
Reply-to: The Simple Times <st-editorial@simple-times.org>
To:       The Simple Times Subscribers <st-editorial@simple-times.org>
Dcc:      st-postscript-subscribers
Fcc:      outbox, simple-times/1.4, /var/lists/simple-times/postscript
Subject:  The Simple Times, volume 1, number 4

#application/postscript [The Simple Times] stv1n4.ps
```

5 Setting Up For MIME

Your system administrator has probably set up some reasonable `mhn` defaults for composing and showing multi-media e-mail messages. These defaults may or may not be adequate for the kinds of multi-media e-mail messages that you receive or compose.

To compose multi-media e-mail messages using pre-existing files requires no special setup; it's displaying them that definitely requires setup.

When you display a message by using `show` or `mhn -show`, you might see error messages like this one:

```
mhn: don't know how to display content
     (content video/mpeg in message /home/jsweet/Mail/drafts/15)
```

A "don't know how to display" error message like the one above means that `mhn` hasn't been informed how to display a particular type of message part.

This problem may or may not be easy to fix, depending on whether you have a program around that can display the particular type of data involved.

`Mhn` looks in several places for information about how to compose or to display multi-media message parts. Your `.mh_profile` is one such place.

The `.mh_profile` file in your home directory exists to permit customizations of the behaviors of MH programs. Many MH users never bother changing the defaults. However, for using `mhn`, changing the `.mh_profile` is almost unavoidable. However, it may not be necessary to change your `.mh_profile` if you're happy with the system defaults.

By using an appropriate display program in an appropriate profile entry, `mhn` can be informed how to run a program to display a particular type of message part.

For example, `mhn` can be instructed to use the `xv` program to display `image/gif` data on an X Window System display. If you don't have the `xv` program, it's certainly possible to obtain it and to compile it for yourself; it's freely available via anonymous FTP from a number of Internet sites. However, you may have some other program on your system capable of displaying GIF files, in which case you could use that program instead of `xv`.

Profile entries can be overridden in message drafts by using appropriate program invocations in the place of file names in the `mhn` directives.

Information about how to obtain various programs that can display multi-media message parts can be found in appendix E. It may also be useful to track down a highly informative document in the MH source tree, `miscellany/multimedia/READ-ME`.

5.1 A Simple Setup

For a plain 24×80 ASCII character terminal, or a plain shell window with no special X Window System display configuration, here are some reasonable `mhn`-related lines to place in your `~/.mh_profile` file:

```
Comment-100: ================================================
```

```
            Global mhn-related parameters:
            ================================================
lproc:                          show
mhnproc:                        mhn
mhn:                            -list -nomoreproc
mhn-storage:                    /usr/tmp
Comment-200: ================================================
            File name templates for storing certain types of
            message parts:
            ================================================
mhn-store-text:                 %m%P.txt
mhn-store-application/PostScript: %m%P.ps
Comment-300: ================================================
            Shell commands for mhn to invoke when mhn
            encounters certain types of message parts:
            ================================================
mhn-show-text/plain:            more '%F'
mhn-show-message/rfc822:        show -file '%F'
mhn-show-text/richtext:         richtext -p '%F'
```

Another profile item that you may find useful is this:

```
Comment-400: ================================================
            Instructions to whatnow(1):
            ================================================
automhnproc:                    mhn
```

Each line beginning with `mhn-` has meaning for a different `mhn` function:

- `mhn-compose-` is the prefix for operations to apply when composing messages;
- `mhn-show-` is the prefix for operations to apply when displaying messages;
- `mhn-store-` is the prefix for operations to apply when storing messages.

What follows any of these `mhn` function prefixes is a *type*. When `mhn` finds a message part of a particular type, `mhn` looks for profile entries relevant to that type. For example, to display something of type `text/richtext`, `mhn` looks for the profile entry `mhn-show-text/richtext`.

The list of pre-defined types can be found in appendix B.

Here is an explanation of each line of the profile:

`lproc: show` —specifies that the `show` program is to be used by `whatnow` when `list` is specified at the "`What now?`" prompt.

`mhnproc: mhn` —specifies that `mhn` is to be used by MH programs for displaying MIME messages. In general, `show` automatically uses `mhn -show`.

`mhn: -list -nomoreproc` —specifies that `mhn` is always to list the contents of a MIME message whenever it's invoked by `show` or in any other context, and that a pager program (actually, an *additional* pager program), such as `more`, shouldn't be invoked.

`mhn-storage: /usr/tmp` —specifies that when `mhn -store` is used, the resulting files (created by extracting the individual message parts from the message) are to be deposited in the `/usr/tmp` directory. You might prefer to specify a directory under your own home directory instead.

`mhn-store-text: %m%P.txt` —specifies that when `mhn -store` is used, and `mhn` finds a message content of type `text`, that the file name should have the form *message-number*.*part-number*.`txt`.

`mhn-store-application/PostScript: %m%P.ps` —specifies that when `mhn -store` is used, and `mhn` finds a message part of type `application/PostScript`, that the file name should have the form *message-number*.*part-number*.`ps`.

`mhn-show-text/plain: more '%F'` —specifies that when `mhn -show` is used, and `mhn` finds a message part specifically of type `text/plain`, that the `more` program should be used to display that part.

The "%F" escape code tells `mhn` to store the text into a temporary file and then to give the name of the temporary file to the `more` program.

Some persons prefer the `less` program to the `more` program for showing plain text.

`mhn-show-message/rfc822: show -file '%F'` —specifies that when `mhn -show` is used, and `mhn` finds a message part of type `message/rfc822`, that the `show` program should be used to display it.

The "%F" escape code tells `mhn` to store the message into a temporary file and then to give the name of the temporary file to the `show` program.

`mhn-show-text/richtext: richtext -p '%F'` —specifies that when `mhn -show` is used, and `mhn` finds a message part of type `text/richtext`, that the `richtext` program should be used to display it.

The "%F" escape code tells `mhn` to store the text into a temporary file and then to give the name of the temporary file to the `richtext` program.

If the `richtext` program isn't present on your system, you can try substituting `rt2raw`. (See appendix E for information about how to obtain these programs.)

`automhnproc: mhn` —tells whatnow(1) that drafts containing `mhn` directives should be run through `mhn` automatically, without waiting for the user to type `edit mhn` at the "`What Now?`" prompt.

5.2 Setting Up For An X Window System Display

For an X Window System display, it's better to create a separate `mhn`-specific profile in addition to your `.mh_profile`, and to refer to that separate profile with the environment variable `$MHN`. This allows you to keep different profiles for displays with different capabilities.

One simple way to set the MHN environment variable for different displays is to use this command:

```
% setenv MHN ~/.mhn_profile.`hostname`
```

This causes `mhn` to look for its defaults in the named file. The `` `hostname` `` part in backticks turns into the name of your host.

If you're using an X terminal separate from your host, then you'll need to use the name of the X terminal instead.

If it hasn't been done for you already, you need to initialize your environment to use the X Window System. Methods vary from system to system, but might be as simple as typing this command:

```
% setenv DISPLAY `hostname`:0
```

...or this:

```
% xinit
```

If your host is named `fungus`, and the `$MHN` variable is set as shown in a previous example above, then you can define an `mhn` profile in your home directory named `.mhn_profile.fungus`. The contents of that file might be specific to the console display of a Sun SPARCstation IPC, designated as X Window System display `fungus:0`, and could look like this:

```
mhn-show-image: xv -geometry =-0+0 -imap -igeom =-0+0 '%f'
mhn-compose-audio/basic:         audiotool '%f' && raw2audio -F < '%f'
mhn-compose-audio/x-next:        audiotool '%f' && adpcm_enc < '%f'
mhn-show-audio/basic:            raw2audio 2>/dev/null | play
mhn-show-audio/x-next:           audiotool -p '%f'
mhn-show-application/PostScript: pageview '%F'
mhn-show-text/richtext:          richtext -p '%F'
mhn-show-video/mpeg:             mpeg_play -quiet -loop '%F'
mhn-cache:                       /r/mhn-cache
```

The programs specific to the Sun SPARCstation in the example above are the audio-related programs, `raw2audio`, `play`, `adpcm_enc`, and `audiotool`; and the PostScript display tool, `pageview`.

Which audio programs to use and how to set up to use them is highly dependent on the type of workstation you have. An explanation is given in appendix D.

Here is an explanation of each line in the previous `mhn` profile:

`mhn-show-image: xv -geometry =-0+0 -imap -igeom =-0+0 '%f'` —specifies that the `xv` program with the specified options is to be used to display all message parts of type `image` (including, for example, `image/gif`).

The "%f" escape code tells `mhn` to store the image data into a temporary file and to give the name of the temporary file to the `xv` program.

`mhn-compose-audio/basic: audiotool '%f' && raw2audio -F < '%f'` —specifies that the `audiotool` program is to be used when a "bare" `#audio/basic` directive (without a filename) is encountered in a message draft. The result is

run through the `raw2audio` program to strip off the audio header information. (Note that these programs are specific to Sun SPARCstations.)

`mhn-compose-audio/x-next: audiotool '%f' && adpcm_enc < '%f'` —specifies that the `audiotool` program is to be used when a "bare" `#audio/x-next` directive (without a filename) is encountered in a message draft. The result is run through `adpcm_enc` program to convert the data into ADPCM format. (An explanation of `adpcm_enc` is given in appendix D. Note that these programs are specific to Sun SPARCstations.)

`mhn-show-audio/basic: raw2audio 2>/dev/null | play` —specifies that when `mhn -show` is used, and a message part of type `audio/basic` is encountered, that the `raw2audio` program is to be used to process the audio data. Any diagnostic or error message are redirected to the bit bucket. The result is piped into the `play` program.

`Mhn` provides the audio data from the message part to `raw2audio`'s standard input.

`mhn-show-audio/x-next: audiotool -p '%f'` —specifies that when `mhn -show` is used, and a message part of type `audio/x-next` is encountered, that the `audiotool` program is to be used to play the audio data.

The "%f" escape code tells `mhn` to store the image data into a temporary file and to give the name of the temporary file to the `audiotool` program.

`mhn-show-application/PostScript: pageview '%F'` —specifies that when `mhn -show` is used, and a message part of type `application/PostScript` is encountered, that the `pageview` program is to be used to display results of running the PostScript program.

The "%F" escape code tells `mhn` to store the PostScript data into a temporary file and to give the name of the temporary file to the `pageview` program.

A publicly available program, GhostScript, aka `gs`, can be substituted.

Note that there are some security issues to consider when deciding to run PostScript processors automatically. It's possible for someone to encode a MIME message to do evil things to an unwary mail user.

PostScript is a big potential security hole. One famous example is the "melting screen" PostScript program, which destroys screens maintained by Display PostScript implementations. For another example, PostScript can be used to change the password on some PostScript printers with previously

undefined passwords, which denies the use of the printer until the printer's password can (somehow) be changed back. Yet other Display PostScript implementations may allow file operations.

The enumeration of these security holes is not to be interpreted as encouragement to exploit the holes. They are mentioned only because they are well known. Refer to books such as *Practical UNIX Security* and to news groups such as `comp.security.misc` for general information about system security.

`mhn-show-text/richtext: richtext -p '%F'` —specifies that when `mhn -show` is used, and a message part of type `text/richtext` is encountered, that the `richtext` program is to be used to display the richtext data.

The "%F" escape code tells `mhn` to store the richtext data into a temporary file and to give the name of the temporary file to the `richtext` program.

`mhn-show-video/mpeg: mpeg_play -quiet -loop '%F'` —specifies that when `mhn -show` is used, and a message part of type `video/mpeg` is encountered, that the `mpeg_play` program is to be used to display the MPEG data.

The "%F" escape code tells `mhn` to store the video data into a temporary file and to give the name of the temporary file to the `mpeg_play` program.

`mhn-cache: /r/mhn-cache` —specifies in what directory to cache external body parts when the parts are retrieved from the remote host. (Note: not all MIME messages have the information necessary to cause `mhn` to cache external body parts. However, any MIME messages created with `mhn` do have the necessary information.)

As you may have guessed from the "pipe" symbols and other UNIX shell metacharacters in the example above, `mhn` invokes `/bin/sh` to interpret the commands.

The difference between "%F" (upper case) and "%f" (lower case) elsewhere is that for "%F," `mhn` runs only one program at a time, and leaves the program's standard input connected to the terminal.

There are other "%" escape codes used by `mhn`. These are described in section 7.

5.3 Examples of mhn_defaults

The defaults for the ways in which mhn displays and stores multi-media e-mail messages can be found in the file mhn_defaults, located in the MH library directory. To find out which directory is the MH library directory, you can type this command:

```
% mhparam mhlproc
```

This prints out the path of the mhl program. In MH 6.8, mhl is in the MH library directory. The mhn_defaults file is in the same directory.

For your perusal, here are some example mhn_defaults files in use at UC Irvine's ICS Department and at Irvine Compiler Corp.

Any of the lines in these files can be picked up and modified for inclusion in your .mh_profile, or in your mhn profile named by the environment variable $MHN.

The various "%" escape codes used in these profiles are described in section 7.

UCI ICS's mhn_defaults:

```
mhn-charset-iso-8859-1: xterm
                        -fn '-*-*-medium-r-normal-*-*-120-*-*-c-*-iso8859-*'
                        -e %s
mhn-show-application/PostScript:  %plpr -Pps
mhn-show-image:                   %p/usr/bs/X11R5/bin/xv -geometry =-0+0 '%f'
mhn-show-text/richtext:           %p/usr/local/lib/mh-6.8/rt2raw '%f'
mhn-store-application/PostScript: %m%P.ps
mhn-store-text:                   %m%P.txt
```

Note that in the absence of a subtype indicator, the line applies to all subtypes. The line for mhn-show-image applies to all image types, such as image/gif and image/jpeg.

ICC's mhn_defaults for Sun Microsystems SPARCstations:

```
mhn-compose-audio/basic:          /usr/local/bin/rec.sparc |
```

```
                                    /usr/demo/SOUND/raw2audio -F
mhn-compose-audio/x-next:           /usr/local/bin/rec.sparc
mhn-show-application/PostScript:    %plpr -Pps
mhn-show-audio/basic:               %p/usr/demo/SOUND/raw2audio
                                    2>/dev/null | play
mhn-show-audio/x-next:              %p/usr/demo/SOUND/play
mhn-store-application/PostScript:   %m%P.ps
mhn-store-audio/basic:              | /usr/demo/SOUND/raw2audio -e ulaw
                                    -s 8000 -c 1
                                    > %m%P.au
mhn-store-audio/x-next:             %m%P.au
mhn-store-text:                     %m%P.txt
mhn-cache:                          /r/mhn-cache
```

The `rec.sparc` program, a small perl script, is described further in appendix D.

ICC's `mhn_defaults` for HP 9000/700 series workstations:

```
mhn-compose-audio/basic:            recorder '%F' -u -pause
mhn-compose-audio/x-next:           recorder '%F' -au -pause
mhn-show-application/PostScript:    %plp -dpsjetdup
mhn-show-audio/basic:               splayer -u
mhn-show-audio/x-next:              splayer -au
mhn-show-image:                     %pxv -geometry =-0+0 '%f'
mhn-show-message/rfc822:            show -file '%F'
mhn-show-text/richtext:             richtext -p '%F'
mhn-show-text/plain:                more '%F'
mhn-store-application/PostScript:   %m%P.ps
mhn-store-audio/basic:              %m%P.u
mhn-store-audio/x-next:             %m%P.au
mhn-store-text:                     %m%P.txt
mhn-cache:                          /r/mhn-cache
```

6 MIME and MH

The ability to create e-mail messages with audio and other non-textual contents has been around for a while, but almost always as part of a vendor-specific "solution." This means that you can't create a message under a

NeXTstep system with PostScript information and *Lip Service* (NeXT's audio e-mail) and directly handle the same message on an HP 9000/710, a Sun SPARCstation IPC, and a Silicon Graphics Iris. That's a problem that MIME helps to solve.

MIME, the Multi-purpose Internet Mail Extensions, is a freely available specification that offers a way to interchange multi-media e-mail among many different computer systems. MIME supports not only several pre-defined types of non-textual message contents, such as μ-law audio, GIF files and PostScript images, but also permits you to define your own types of message parts.

The MIME specification is given in the Internet "Request for Comments" document RFC 1341. It's one of the more readable RFC documents, so if you're interested in the details of MIME, I suggest that you find a copy and read it. (See section E for information about how to obtain copies.)

MIME isn't specific to MH; there are other mail systems that support MIME. MH offers one widely available implementation of the MIME specification.

If you're curious about what the MIME representation of a message looks like, try typing `list -noshowproc` after you've run `edit mhn` on a draft and before you've sent it:

```
What now? list -noshowproc
```

It's also possible to view raw MIME by using a command like this one:

```
% show -noshowproc
```

7 Mhn Explained

The manual page mhn(1) describes `mhn` in concise detail. The manual page should always be consulted for the most up-to-date information about `mhn`'s features. This section provides a simplified synopsis of those features.

The mhn(1) manual page uses the terms *content* and *content type* to refer to the data in a multi-media message part and to the type of that data. For simplicity, this tutorial has used the term *message part* instead.

The `mhn` program allows you to create and to view MIME messages in just about any way that meets your needs, using whatever programs are at hand or whatever programs you invent.

The default configuration, as installed by your system administrator, most likely comes with some useful means by which to compose and to display the most common types of multi-media message parts.

The `mhn` program can be invoked directly from the command line to manipulate MIME messages, using the same kinds of options that other MH commands use:

- The `-list` option enumerates the contents;
- The `-show` option displays the contents;
- The `-store` option stores the contents into files.

The synopsis of the `mhn` command from its manual page is this:

`mhn` [`+`*folder*] [*msgs*] [`-part` *number*]... [`-type` *content*]...

```
    [ -list [ -headers ] [ -noheaders ]
            [ -realsize ] [ -norealsize ] ]
            [ -nolist ]

    [ -show [ -serialonly ] [ -noserialonly ] ]
            [ -form formfile ] ]
            [ -noshow ]

    [ -store [ -auto ] [ -noauto ] ]
            [ -nostore ]

    [ -verbose ] [ -noverbose ]
    [ -rfc934mode ] [ -norfc934mode ]
```

[-ebcdicsafe] [-noebcdicsafe]
[-help]

Some, but not all, of the options are explained here; for more information, consult mhn(1).

Typically, the MH command `show` is configured with an `mhnproc` parameter in the `.mh_profile` so that it can automatically invoke `mhn`. (See section 5.)

Either an entire message can be displayed, or just an individual part. To select an individual part, use the `mhn -part` option, as in this example:

```
% mhn -part 1.3 -show
```

7.1 Profile Entries

For each of the different `mhn` functions, `mhn` looks for certain kinds of profile entries. `Mhn` obtains profile information in this order of preference:

1. From a file named by the `$MHN` environment variable. This allows you to keep different multi-media message handling profiles for each type of display or workstation that you use. Section 5 gives examples of how to do this.
2. From your `.mh_profile`, or in the profile specified by the `$MH` environment variable.
3. From the MH library file `mhn_defaults` (created by your system administrator).
4. From some built-in defaults for a few simple content types, such as `text/plain`.

Here is a list of profile entries for which `mhn` looks:

`mhlproc`: The program to display message headers when `mhn-show` is used. The default is `mhl`.

`mhn-access-ftp`: The program to retrieve message parts via FTP when `mhn -show` is used with a message containing external parts. The default is to use a built-in FTP retrieval mechanism.

`mhn-cache`: The directory in which to store cached external parts when `mhn -show` is used with a message containing external parts. The default is not to cache external parts at all, but rather to retrieve them each time. (See section 8.)

`mhn-charset-`*charset*: A shell command to create environment to render particular kinds of character sets when `mhn -show` is used. For example, `xterm` can be run with an appropriate font designation, or the `less` program can be run with appropriate environment variable settings.

`mhn-compose-`*type*: A shell command to compose message parts on the fly when `edit mhn` is used from the `What now?` prompt.

`mhn-show-`*type*: A shell command to display message parts of the specified type when `mhn -show` is used.

`mhn-storage`: The directory in which to store message parts when `mhn -store` is used. The default is the current working directory.

`mhn-store-`*type*: A file name template or a shell command (beginning with the pipe symbol "`|`") for storing message parts of the specified type when `mhn -store` is used.

`moreproc`: The program to display `text/plain` message parts. The default is `more`.

7.2 Displaying Messages

The `mhn -show` command can be used to display multi-media messages.

Alternatively, the `show` command should use `mhn` by default. If it doesn't, try using this profile entry:

```
mhnproc: mhn
```

Some examples of `mhn-show-` profile entries are:

`mhn-show-audio/basic` —Audio setups can be tricky. See appendix D for a detailed explanation. This example is for HP 9000/710 workstations. Your audio playing command may differ.

```
mhn-show-audio/basic: splayer -u
```

`mhn-show-image` —Specifying a profile entry for `image` with no subtype means that the entry applies to all subtypes. For example, the `xv` program handles both `image/gif` and `image/jpeg` types, as well as a number of others:

```
mhn-show-image: %pxv -geometry =-0+0 '%f'
```

`mhn-show-application/postscript` —Messages containing `application/postscript` parts can be viewed using the freely available GNU GhostScript tools, `gs` and `ghostview`, or using vendor-supplied PostScript viewing tools, such as Sun's `pageview`. Alternatively, PostScript parts can be viewed by printing them on PostScript printers. Any of these examples might do:

```
mhn-show-application/PostScript: %ppageview '%F'
mhn-show-application/PostScript: %pgs -
mhn-show-application/PostScript: %pghostview -
mhn-show-application/PostScript: %plp -dpsjetdup
```

Note the security issues surrounding PostScript raised in section 5.2.

Shell commands in `mhn-show-` profile entries may include *escape codes* beginning with a percent sign ("%"). The "%" escape codes include these:

`%f` The "%f" escape tells `mhn` to store the data into a temporary file and to give the name of the temporary file to the display program.

`%F` The "%F" escape tells `mhn` to store the data into a temporary file and to give the name of the temporary file to the display program.

The difference between %F (upper case) and %f (lower case) is that for %F, `mhn` runs only one program at a time, and leaves the program's standard input connected to the terminal.

Note that when using the %f or %F escapes, it's a good idea to use single quotes around the escape, as in `'%F'`. This prevents misinterpretation by the shell of any funny characters that might be present in the filename, such as `<` and `>`, which occur in the names of cached external body part files.

`%p` The "%p" escape specifies that the part's description is to be listed and then a prompt delivered to ask whether or not the part should be shown.

When a %p causes a prompt for confirmation, typing INTR (usually control-C) tells mhn not to display that part. Typing QUIT (usually control-\) aborts the entire message display process.

The mhn(1) manual page lists other "%" escapes that may be useful in `mhn-show-` profile entries.

7.3 Composing Messages

MH 6.8 doesn't provide a visual front end for creating MIME messages. This means that you need to type some simple directives into your message draft, which takes a little getting used to if, for example, you've been using NeXTmail.

`Mhn` can be invoked from the "What now?" prompt to create a MIME message:

```
What now? edit mhn
```

Using `edit mhn` invokes the "compose" function of `mhn`.

It's also possible to instruct `whatnow` (the MH program responsible for handling your "`What now?`" instructions) to invoke `mhn` automatically, so that you don't need to type `edit mhn`. To do that, use an entry like this in your `.mh_profile`:

```
automhnproc: mhn
```

When `mhn` is invoked to compose a MIME message, each `#` directive in the message draft causes data from a file or from a command to be inserted.

If a file isn't specified with the directive, then a command is used to generate the data. For example, consider this directive:

```
#audio/basic [record something here]
```

Because there's no file specified, `mhn` looks for this profile entry to record `audio/basic` data to insert into the draft:

```
mhn-compose-audio/basic: recorder '%F' -u -pause
```

This example is for HP 9000/710 workstations. Your audio recording command may differ. Refer to appendix D for more information about audio tools.

Two escape codes relevant to composition commands are these:

`%f` The "%f" escape tells `mhn` to supply the name of a temporary file, from which the data are taken when the program ends.

`%F` The "%F" escape tells `mhn` to supply the name of a temporary file, from which the data are taken when the program ends.

The difference between %F (upper case) and %f (lower case) is that for %F, `mhn` doesn't redirect standard output (typically so that it's left attached to the terminal).

Other escape codes that may be of use are documented in mhn(1).

In an `mhn` directive in a message draft, an appropriate program invocation (beginning with the pipe symbol, "`|`") in the place of a file name overrides an `mhn-compose-` profile entry.

Here is a synopsis of the syntax of `mhn` directives:

$$
\begin{array}{lcl}
\textit{body} & \rightarrow & (\textit{content} \mid \textit{EOL})^{+} \\
\textit{content} & \rightarrow & \textit{directive} \mid \textit{plaintext} \\
\textit{directive} & \rightarrow & \texttt{\#}\ \textit{type}\ \texttt{/}\ \textit{subtype} \\
 & & \quad (\texttt{;}\ \textit{attribute}\ \texttt{=}\ \textit{value})^{*} \\
 & & \quad [\ (\ \textit{comment}\ )\ ]
\end{array}
$$

[[*description*]]
[*filename*]
EOL
—For files or programs

|

#@ *type* / *subtype*
(; *attribute* = *value*)*
[(*comment*)]
[[*description*]]
external-parameters
EOL
—For external body parts

|

#forw
[[*description*]]
[**+***folder*] [*msg**]
EOL
— For message digests

|

#begin
[[*description*]]
[**alternative** | **parallel**]
EOL
body+
#end *EOL*
—For multipart messages

plaintext → [**Content-Description:**
description EOL EOL]
line+
[**#** *EOL*]
—Describes and encloses a `text/plain` part

|

#< *type* / *subtype*
(; *attribute* = *value*)*
[(*comment*)]

[[*description*]]
EOL
line$^+$
[**#** *EOL*]
—Encloses plain text representation of data

line → **##** *text EOL*
—Interpreted as **#***text EOL*
|
text EOL

Examples of various forms of `mhn` directives are scattered throughout this tutorial, particularly in sections 4, 8, and appendix B.

Section 8 has a list of required *attribute*=*value* pairs to apply to external types. The MIME specification, RFC 1341, contains more detailed explanations of required or optional attributes. Type names and attributes used with `mhn` directives correspond to those used in the MIME specification.

7.4 Storing Message Parts

The `mhn -store` option stores message parts (all parts if `-part` isn't specified) in "native" (decoded) format.

Here is example of an `mhn -store` command:

```
mhn 11 -part 3.3 -store
```

This stores part 3.3 of message number 11 into a file. The name and location of the file may be specified by profile entries like these:

```
mhn-storage:                        /usr/tmp
mhn-store-application/PostScript: %m%P.ps
mhn-store-audio/basic:              %m%P.u
mhn-store-audio/x-next:             %m%P.au
mhn-store-text:                     %m%P.txt
```

The `mhn-storage` profile entry specifies the directory in which to store files. The default is the current working directory.

In `mhn-store-` profile entries, either a file name formatting string or a program invocation (beginning with the shell pipe symbol, "`|`") may be specified, as in this example:

```
mhn-store-audio/basic: | /usr/demo/SOUND/raw2audio
                           -e ulaw -s 8000 -c 1
                           > %m%P.au
```

This example is for Sun SPARCstations. The command runs the audio data through a program that adds descriptive information to the audio data before storing it into a file. A more comprehensive explanation of what's going on is given in appendix D.

Escape codes beginning with "%" specify what the file names are to look like.

Given the command at the beginning of this section and the previous profile entry for storing `audio/basic` parts, the `%m%P.au` specification refers to a file named `11.3.3.au`.

These are the escape codes for `mhn-store-` profile entries:

`%m`	message number
`%P`	.part
`%p`	part
`%s`	subtype

There are other file name formatting options described in mhn(1).

7.5 Other Facilities

`Mhn` has a number of other facilities for manipulating MIME messages. Detailed information about how to use the facilities can be found in the mhn(1) manual page; here is a brief list of some of them:

Splitting large messages. MIME permits a message to be split among several separate messages. The `send -split` option automatically splits a message. The `mhn -store` option can be used to reassemble a split message. (Split messages, each of which are *partial* messages, are those using the MIME type `message/partial`.)

Retrieving external body parts. Some examples of external bodies have been given in section 4.1. More examples are given in section 8. There are several different ways to refer to and to retrieve external body parts. One alternative, using mail server references, is explained in section 8.

Sending file sets. There is a script, `viamail`, available in the MH library directory, which sends a set of files via e-mail. The files are automatically archived into a compressed UNIX tar file format (`.tar.Z`) and transported in a MIME message (or in several messages, split automatically, if the archive is large). The `mhn -store` option is also capable of extracting the files from such MIME messages.

Encoding and extracting arbitrary data. The MIME "base64" encoding format is a fairly safe way to transport data (such as shell scripts and program binaries) through mail gateways. By specifying the `application/octet-stream` type when composing a message, `mhn` will encode in base64 any file you specify. MIME-capable user agents should be able to decode that file. In MH, `mhn -store` can be used to decode an `application/octet-stream` message part encoded in base64.

Setting up to use `less`. `Mhn` will use the `less` pager program instead of `more` if you so specify in the `moreproc:` profile entry. Some additional environment setup is required, as described in the mhn(1) manual page.

8 More About External Body Parts

External body parts—particularly when cached on local systems—have the advantage that if there are many recipients, less network bandwidth and disk storage may be consumed than when the complete image or audio contents are included in every copy of the message.

However, there are some potential pitfalls to using external body parts:

1. Some recipients of `access-type=anon-ftp` parts might not have Internet access. Internet FTP per se isn't possible for non-Internet sites.

2. A file on a remote host might not be accessible at all times because of transient network problems (an all-too-frequent occurrence on the Internet).

3. In the case of the `mail.weather.ext` program mentioned in section 4.1, sometimes the image files, whose names are generated automatically based on the date and the time of day, are sometimes not present on the remote host, even hours after the files are expected to be there.

The first limitation can be overcome when files available via anonymous FTP are also available via a *mail server*. A mail server is a program that accepts an e-mail message containing a request for something and returns the requested item to the originator of the request via e-mail. Mail servers are especially useful under circumstances where Internet FTP doesn't work, such as over dial-up UUCP links.

To permit access to a file either via anonymous FTP or via a mail server, enclose two external file references inside a `#begin alternative` directive, as in this example:

```
#begin alternative
#@image/gif [via FTP: surface & radar analysis] \
        access-type=anon-ftp; \
        name="SA022509.GIF"; \
        directory="wx"; \
        site="vmd.cso.uiuc.edu"; \
        expiration="26 Feb 93 09:00:00 GMT"; \
        size=50000
#@image/gif [via mail: surface & radar analysis] \
        access-type=mail-server; \
        expiration="26 Feb 93 09:00:00 GMT"; \
        size=50000; \
        server="wx-server@vmd.cso.uiuc.edu"; \
        body="send wx:SA022509.GIF"
#end
```

A message containing these directives has this structure, as shown by `mhn -list`:

```
msg part  type/subtype              size description
 1        multipart/alternative      697

    1     message/external-body      110
          image/gif                      via mail: surface & radar analysis

    2     message/external-body       87
          image/gif                      via FTP: surface & radar analysis
```

Note that the order of alternatives listed is the reverse of that in which the directives occurred in the original draft.

The message, when viewed by the MH command `show`, results in a session like this one:

```
Retrieve content 1 by asking mailbox wx-server@vmd.cso.uiuc.edu

send wx:SA022509.GIF

? y
mhn: request sent
```

The retrieval message sent by `mhn` looks like this:

```
To: wx-server@vmd.cso.uiuc.edu
Subject: cache content as /r/mhn-cache/<24586.731500909.2@fester.irvine.com>

send wx:SA022509.GIF
```

When the mail server on the remote host receives this message, it takes action based on the command specified in the body. The body may specify any command that the mail server understands. Note that unlike FTP, there is no standardization among different mail servers. The `send` command shown in the example above might need to be a `get` command, or something completely non-intuitive, for a mail server at another site.

(By the way, this particular example, although semantically correct, won't really work, because the site `vmd.cso.uiuc.edu` doesn't offer files via a mail server—not to mention that the particular file named has long since been deleted from the system. FTP is really the only retrieval method that applies for the weather maps.)

If the answer to the question above had been "n" then the usual FTP retrieval question would have been asked:

```
Retrieve SA022509.GIF (content 2)
    using anonymous FTP from site vmd.cso.uiuc.edu (50000 octets)?
```

In general, alternatives permit users to have different ways to look at a message. For those recipients without the necessary resources to view a particular message part, it helps to offer a "plain" alternative—even if it's just to say "sorry, but you're missing something neat..."

RFC 1341 (the MIME specification) recommends that the "plainest" alternative be placed first in the list, and that the "fanciest" alternative be placed last. The MH `show` command offers the last alternative first. In our example above, the `access-type=mail-server` alternative should have gone second, since it doesn't offer gratification as immediate as the `access-type=anon-ftp` alternative does.

Here is a list of the `mhn` directive forms most likely to be used for inserting external body parts:

To refer to an external file available via anonymous FTP:

```
#@type/subtype [description] \
        access-type=anon-ftp; \
        name="remote-filename"; \
        directory="remote-directory"; \
        site="remote-ftp-host"
```

To refer to an external file available via mail server:

```
#@type/subtype [description] \
        access-type=mail-server; \
```

```
        server="mailbox-address"; \
        body="retrieval-command"; \
```

To refer to a locally available file:

```
    #@type/subtype [description] \
        access-type=local-file; \
        name="local-path"
```

There are other kinds of `access-type` attributes that may be used for alternative file access methods. Refer to the mhn(1) manual page and to RFC 1341 for information about these other access types.

Also, you don't have to use `mhn`'s built-in FTP retrieval mechanism. `Mhn` will use an FTP program that you designate in this profile entry:

`mhn-access-ftp:` *my-ftp-program*

Information about the requirements for an `mhn`-invoked FTP retrieval program can be found in the mhn(1) manual page.

A General Information about MH

MH is a modular mail system comprised of a set of shell-level commands.

The MH commands permit you not only to handle your mail in the usual way, e.g. to read messages, to produce scan summaries, and to reply to messages, but also to cobble together your own mail handling utilities by invoking MH commands from shell scripts or perl programs.

Here are two handy papers, supplied with the MH software distribution, that you should seek out and read if you're unfamiliar with MH:

- "The Rand MH Message Handling System: Tutorial" by Marshall Rose and Jerry Sweet

- "How to process 200 messages a day and still get some real work done" by Marshall Rose and John Romine

...And here are two useful books that cover MH:

- *MH & xmh* by Jerry Peek (O'Reilly & Associates, Inc).
- *The Internet Message* by Marshall Rose (P T R Prentice-Hall, Inc.)

Peek's book, as of the 1992 edition, doesn't cover MH's MIME facilities, although it's a good reference book for MH users. Rose's book isn't specifically about MH, but does include a lengthy discussion of MH's MIME facilities.

See appendix E for information about how to obtain the papers. See appendix G for ISBN information for the books.

B Pre-Defined MIME Types

Each part of a multi-media message identifies what type of information is carried in the message part. For example, a message part containing audio data might have either type `audio/basic` or type `audio/x-next`. Both are `audio` types; the *subtypes* are `basic` and `x-next`.

An entire MIME message—as opposed to an individual part of a multipart message—can also have a type. For example, a message might have the type `text/plain`, and consist entirely of plain text. A MIME message containing parts of different types has the umbrella type `multipart/mixed`.

Here are brief summaries of the pre-defined types and subtypes in MIME 1.0. Except where noted, each of these types may be specified in an `mhn` directive in a message draft, preceded by a hash mark (`#`) at the start of a line.

`application/octet-stream` is for "other" kinds of data, either uninterpreted binary data or information to be processed by a mail-based application. When `mhn` incorporates the data directly into a draft, the data is encoded in MIME's base64 format. This is also a good way to send data, such as

shell scripts, that might suffer unacceptable modifications (e.g. long text line wraps) in transport.

`application/oda` is for ODIF data (open document interchange format, ISO/IEC 8613). When `mhn` incorporates the data directly into a draft, the data is encoded in MIME's base64 format.

`application/postscript` is for PostScript programs (typically documents represented in PostScript form). `Mhn` leaves PostScript data alone when incorporating it into a draft.

`audio/basic` is for audio data encoded using 8-bit ISDN μ-law (Pulse Code Modulation). A sample rate of 8000/second and a single channel is assumed. (This is the "native" format for SPARCstations and for HP 9000/700s.) When `mhn` incorporates the audio data directly into a draft, the data is encoded in MIME's base64 format.

`image/gif` is for GIF (graphical interchange format) image data. When `mhn` incorporates the image data directly into a draft, the data is encoded in MIME's base64 format.

`image/jpeg` is for JPEG (joint picture experts group) image data. When `mhn` incorporates the image data directly into a draft, the data is encoded in MIME's base64 format.

`message/external-body` is for specifying large bodies by reference to an external data source. An `mhn` type specification directive beginning with `#@` specifies a `message/external-body` part.

`message/partial` is for partial messages, to permit the fragmented transmission of bodies that are thought to be too large to be passed through mail transport facilities. Using the `-split` option with `send` creates messages of type `message/partial`.

`message/rfc822` is an encapsulated RFC 822 conformant message which may have its own type. The `mhn` directive `#forw` uses the `message/rfc822` type in conjunction with the `multipart/digest` type.

`multipart/alternative` represents the same data in multiple formats. The `mhn` directive `#begin alternative` uses the `multipart/alternative` type.

`multipart/digest` is for multipart entities in which each part is an encapsulated message. The `mhn` directive `#forw` uses the `multipart/digest` type.

`multipart/mixed` is the primary way to represent a MIME message containing parts of various different types. With `mhn`, a draft (or a part of a draft) containing more than one part is created using the `multipart/mixed` type.

`multipart/parallel` is for data intended to be viewed simultaneously. The `mhn` directive `#begin parallel` uses the `multipart/parallel` type.

`text/plain` indicates plain (unformatted) text.

`text/richtext` is plain text with formatting commands to enhance the appearance of the text. The formatting commands are summarized in appendix C.

`video/mpeg` is for MPEG (motion picture experts group) video data. Video requires the capability to display moving images, typically including specialized hardware and software. When `mhn` incorporates the audio data directly into a draft, the video data is encoded in MIME's base64 format.

There may be other registered types and subtypes down the road. MIME also allows arbitrary subtypes whose names are prefixed with "x-", but anything else is reserved for registered types.

The MIME specification, RFC 1341, contains more detailed explanations of required or optional attributes to be used with particular types. Type names used with `mhn` correspond to those used in the MIME specification.

Base64 is a textual encoding used in MIME messages. Base64 is designed to permit data to survive the most common sorts of transformations that mail gateways apply to message bodies, such as breaking long lines of text into multiple lines. Base64 is similar in concept to the encoding used by `uuencode`, except that uuencoded data doesn't always survive intact when transported through certain mail gateways.

C Richtext

This section summarizes the `text/richtext` type. A full explanation of richtext can be found in RFC 1341, the MIME specification.

Richtext is a simple markup language for plain text, like LaTeX or nroff or Scribe. A document in richtext format consists of plain text "marked up" with formatting commands. The formatting commands are used to change fonts, to format paragraphs, and to lay out pages. For those with some word processing experience, MIME richtext isn't in any way related to Microsoft Richtext.

The formatting commands are words surrounded by angle brackets (<>), such as <bold>. Each occurrence of a command is balanced later on by a negating command beginning with a forward slash or solidus (/), such as </bold>.

There are only three exceptions to this balancing rule:

1. The command <lt>, which represents a literal < character.
2. The command <nl>, which represents a required line break. (Otherwise, CRLFs in the data are treated as equivalent to a single SPACE character.)
3. The command <np>, which represents a page break.

The list of pre-defined richtext commands follows.

Font selection commands:

<bigger> – subsequent text is displayed in a bigger font.
<bold> – subsequent text is displayed in bold font.
<fixed> – subsequent text is displayed in fixed width font.
<italic> – subsequent text is italicized.
<smaller> – subsequent text is displayed in a smaller font.
<subscript> – subsequent text is interpreted as a subscript.
<superscript> – subsequent text is interpreted as a superscript.
<underline> – subsequent text is underlined.

Paragraph formatting commands:

 – subsequent text is centered.
<excerpt> – subsequent text is interpreted as a textual excerpt from another source. Typically this will be displayed using indentation and an alternate font, but such decisions are up to the viewer. In one implementation, each

displayed line of the excerpt begins with a right angle bracket (`>`) and some horizontal whitespace.

`<flushleft>` – subsequent text is left justified.

`<flushright>` – subsequent text is right justified.

`<indent>` – subsequent text is indented at the left margin.

`<indentright>` – subsequent text is indented at the right margin.

`<nl>` – causes a line break. No balancing `</nl>` is allowed.

`<outdent>` – subsequent text is outdented at the left margin.

`<outdentright>` – subsequent text is outdented at the right margin.

`<paragraph>` – subsequent text is interpreted as a single paragraph, with appropriate paragraph breaks (typically blank space) before and after.

`<signature>` – subsequent text is interpreted as a *signature*. Some systems may wish to display signatures in a smaller font or otherwise set them apart from the main text of the message.

Page layout commands:

`<footing>` – subsequent text is interpreted as a page footing.

`<heading>` – subsequent text is interpreted as a page heading.

`<np>` – causes a page break. No balancing `</np>` is allowed.

`<samepage>` – subsequent text is grouped, if possible, on one page.

Character set selection commands:

`<iso-8859-`*X*`>` (for any value of *X* that is legal as a `charset` parameter) – subsequent text is interpreted as text in the appropriate character set.

`<us-ascii>` – subsequent text is interpreted as text in the US-ASCII character set.

Miscellaneous commands:

`<comment>` – subsequent text is interpreted as a comment, not to be shown to the reader.

`<lt>` – replaced by a literal `<` character. No balancing `</lt>` is allowed.

`<no-op>` – has no effect on the subsequent text.

Each positive formatting command affects all subsequent text until the matching negative formatting command. Such pairs of formatting commands must be properly balanced and nested. Thus, a proper way to describe text in bold italics is:

```
<bold><italic>the-text</italic></bold>
```

or, alternatively,

```
<italic><bold>the-text</bold></italic>
```

but, in particular, the following is illegal richtext:

```
<bold><italic>the-text</bold></italic>
```

No special behavior is required for the Tab (aka HT or control-I) character, but when fixed-width fonts are in use, the common semantics of the TAB character may be observed, i.e. that a Tab moves to the next column position that is a multiple of 8.

A line break (CRLF) is interpreted as a *soft* line break, which is treated as horizontal white space. To include a *hard* line break (one that must be displayed as such), the `<nl>` or `<paragraph>` formatting constructs should be used.

Consider this richtext body fragment:

```
<bold>Now</bold> is the time for
<italic>all</italic> good men
 <smaller>(and <lt>women>)</smaller> to
<ignoreme></ignoreme> come

to the aid of their
<nl>
beloved <nl><nl>country. <comment> Stupid
quote! </comment> -- the end
```

The example fragment above represents this formatted text:

Now is the time for *all* good men (and <women>) to come to the aid of their
beloved

country. -- the end

Note that the final display of a message in richtext format may be quite different from that which was intended, simply because display devices and richtext implementations may differ in capabilities. A richtext message that may display beautifully with full italic and bold characters on an X Window System display may look terrible on a 24-by-80 fixed-font ASCII terminal that can only show plain or bold text blocks.

Richtext message parts are specified with the type `text/richtext`. To compose a message with richtext parts, these `mhn` directives may be used:

To insert a text file containing richtext formatting commands:

```
#text/richtext filename
```

To compose a portion of a draft with richtext:

```
#<text/richtext
lines of text with richtext formatting commands
#
```

Of course, external body parts may also be specified with the directive `#@text/richtext` followed by appropriate parameters (see section 8).

D Audio Setup

Much of the following information is taken primarily from a document in the MH 6.8 source tree, `miscellany/multimedia/READ-ME`. You may wish to consult that document to see if it contains more recent information.

MIME pre-defines one audio type, `audio/basic`. This is for audio data encoded using 8-bit ISDN μ-law (Pulse Code Modulation). A sample rate of 8000/second and a single channel is assumed. This is the "native" format for the audio chips included with SPARCstations and for HP 9000/7xx series workstations. References to "u-law" or "mu-law" in other documents mean μ-law.

There are many different types of audio formats, some of which may use compression, multiple channels, and higher or lower sample rates. These

types may also be used in MIME messages by using unofficial `x-` subtypes, such as `audio/x-next`. However, there must exist pre-agreement among users of `x-` subtypes as to what their meanings are and how they are handled.

Which audio programs to use and how to set up to use them is highly dependent on the type of workstation you have. Information is presently available for two types of workstations: Sun SPARCstations and HP 9000/710s. If other audio-capable workstations come with programs that can be instructed from the command line to record or to play particular files, then these programs can be used with MH one way or another.

If your workstation doesn't come with a built-in microphone, or an external microphone supplied by the vendor, you'll need to buy one. If the microphone doesn't have enough gain, you may have to keep the microphone close to your lips when recording. The SONY ECM-K7 microphone comes highly recommended; you can put the microphone on your desk and it will still pick up your voice.

D.1 SPARCstation

Most newer models of SPARCstations have a telephone-quality audio chip and an internal speaker. The "native" encoding for the audio chip is 8-bit μ-law sampled at 8000/second.

Audio data can be manipulated in two formats: "raw" and "self-describing:"

1. "Raw" audio data is the native format used directly by the audio chip. This corresponds to the `audio/basic` type. Audio files prior to SunOS 4.1 used this format only.

2. "Self-Describing" data consists of a header describing the type of audio data in the file (e.g. native μ-law, ISDN A-law, ADPCM, etc.), followed by audio data of that type. This format was introduced with SunOS 4.1.0. The `audio/x-next` type can be used to refer to this self-describing type (so named because NeXT developed this format; Sun supports a subset of it).

A file containing raw audio data can be played by sending it directly to the audio device, `/dev/audio`, as with this command:

```
% cat audio.raw > /dev/audio
```

The header must be stripped from self-describing audio data when specifying an `audio/basic` message part. The header must be present when specifying an `audio/x-next` part.

In support of the `audio/basic` and `audio/x-next` types, it is possible to convert between the raw format and the self-describing format by using a patched version of the `raw2audio` program. The patches are described a little later on.

The `record` program delivers self-describing data. The `play` program plays self-describing data. These programs are found in the `/usr/demo/SOUND/` directory when the demo program package is installed. However, the `record` program is not suitable for running from `mhn`, because the control-C that terminates a recording session raises havoc. The `audiotool` program, if available, is more suitable. In the absence of the `audiotool` program, you can use a little wrapper program for `record` that prompts for the beginning and end of the recording session. An example wrapper, `rec.sparc`, can be obtained as described in appendix E.

The `audio/x-next` type may be used with any self-describing audio data recognized by the SunOS audio programs, such as ADPCM format. The ADPCM format is compressed μ-law encoding based on the CCITT G.721 Adaptive Delta Pulse Code Modulation algorithm. Sun introduced support for this type of audio data with OpenWindows 3.0.

ADPCM format data cannot be directly played to or recorded from the audio chip; it must first be converted back to the native 8-bit μ-law format. Conversions to and from the two encodings can be done with the `adpcm_enc` and `adpcm_dec` programs found in the `$OPENWINHOME/bin/` directory.

Suggested profile entries for `mhn` relevant to recording and playing back audio data in ADPCM format are these:

```
mhn-compose-audio/x-next:       audiotool '%f' && adpcm_enc < '%f'
```

```
mhn-show-audio/x-next:          %paudiotool -p '%f'
```

Suggested profile entries for `mhn` relevant to recording and playing back audio data in native μ-law format are these:

```
mhn-compose-audio/basic: audiotool '%f' && raw2audio -F < '%f'
mhn-show-audio/basic:    %praw2audio 2>/dev/null | play
```

Because the `raw2audio`, `play`, and `record` programs are provided in source form with SunOS 4.1, it is possible to compile them for systems running earlier versios of SunOS. The sources can be found in the SunOS 4.1 Demos installation.

If these programs are unavailable, then you will be able to manipulate only the raw, native audio data formats by reading and writing directly to the `/dev/audio` device via `cat`.

If you do have the `raw2audio`, `play`, and `record` programs (regardless of which version of SunOS you're running), then you need to apply some source changes by using the publicly available `patch` program. The source patches are part of the MH 6.8 source tree:

```
% (cd /usr/demo/SOUND; patch) < miscellany/multi-media/SPARC/SOUND.diff
% cd /usr/demo/SOUND
% make raw2audio
```

These patches add a `-F` option to the `raw2audio` program, and add recognition of types for several different audio data formats.

The `raw2audio -F` option strips the header from self-describing audio data. This enables you to compose message drafts with `#audio/basic`.

Be sure to put the audio programs from `/usr/demo/SOUND` into a directory named in your `$PATH` environment variable so that the audio programs can be found if absolute paths aren't used in the invocations.

Other useful audio programs supplied with later versions of SunOS 4.1 include these:

`/usr/demo/SOUND/x_gaintool` —This program allows you to fiddle with the volume controls whenever you like.

`$OPENWINHOME/bin/xview/audiotool` —This program provides menus and control panels for playing, recording, and editing different types of self-describing sound files.

The directory `/usr/demo/SOUND/sounds/` contains various pre-recorded sound files. The audio files have the extension `.au`.

D.2 HP 9000/710

The HP-Apollo 9000/710 workstation has a telephone-quality audio chip and an internal speaker. Newer workstations in the 9000/700 series may also be equipped for audio; the 9000/720 and 9000/730 models aren't.

HP provides a number of programs in source form that may be used with `mhn`:

- `player` and `splayer`;
- `recorder` and `srecorder`

The directory `/usr/audio/examples/` contains these programs. The `README` file in that directory documents the various audio utility programs for recording and playing sounds.

The programs from `/usr/audio/examples` should be copied into a directory named in your `$PATH` environment variable so that the audio programs can be found if absolute paths aren't used in the invocations.

Before any of the aforementioned programs can be used, the audio device files must be created by running `/usr/audio/bin/make_audio_dev`, and two daemons must be started by the superuser: `/usr/etc/ncs/llbd` and `/usr/audio/bin/Aserver`. Refer to the man page Audio(5) for detailed, step by step instructions on how to do these things. Alternatively, refer to the HP publication "Audio Users Guide" (HP order number A1991-90609, November 1991 version).

The audio devices support several "native" encodings that vary as to which of the audio device files is used: one of `/dev/audio[BEI][ALU]`, where `E`=external jack, `I`=internal speaker, `B`=both; `A`=8-bit A-law, `L`=16-bit linear, `U`=8-bit μ-law. It is possible to `cat` a raw audio data file of the appropriate type to or from one of these devices. For example, a raw SPARC audio file may be played by running this command:

```
% cat bark.raw >/dev/audioIU
```

There is also a raw `/dev/audio` device.

(You must create the audio device files before you can cat anything to them; again, refer to Audio(5) for information on how to do this.)

In HP-UX 8.07, the audio tools, rather than looking for header information in the sound data, rely on user-supplied information, such as preferences, file extensions, or command line options, to inform the tools about the types and sampling rates of the audio data. This means that both `audio/basic` and `audio/x-next` can be used, but you have to use an appropriate file extension or command line option as follows:

type	*extension*	*option*
`audio/basic`	`.u`	`-u`
`audio/x-next`	`.au`	`-au`

The relevant `mhn` profile entries for recording and playing audio parts are these:

```
mhn-compose-audio/basic:  recorder '%F' -u -pause
mhn-compose-audio/x-next: recorder '%F' -au -pause
mhn-show-audio/basic:     splayer -u
mhn-show-audio/x-next:    splayer -au
```

In general, HP audio file names have the form

file name[[.*sample rate*].*type*]

where

sample rate → *digit*$^+$ 000 | *digit*$^+$ k
type → u | al | au | wav | snd | l8 | lo8 | l16

The audio types have these meanings:

μ-law	u
A-law	al
Sun (self-describing format)	au
Microsoft RIFF Waveform	wav
Macintosh	snd
Linear8	l8
Linear8Offset	lo8
Linear16	l16

Examples:

`bark.au` —a Sun-format (`audio/x-next`) sound file

`bark.8000.u` —a raw μ-law sound file recorded at a sample rate of 8000/second.

`bark.8k.u` —same data as in `bark.8000.u`.

As with the SPARC, the default sample rate is 8000/second.

Other useful programs supplied with HP-UX 8.07 are these:

`/usr/audio/bin/audio_demo` —This program provides menus and control panels for playing, recording, and editing different types of sound files. The program also may be used to convert manually among different sound types. Unfortunately, the `audio_demo` program doesn't accept command line arguments, so it isn't quite as suitable for invocation from `mhn` as are the `player` and `recorder` programs.

`/usr/audio/examples/acontrol` —This allows you to fiddle with the volume controls whenever you like.

E On-Line Resources

This section talks about how to obtain some of the documents mentioned in this tutorial.

▷ **RFC 1341** — This is the definition of MIME 1.0. You can obtain it via anonymous FTP from a number of sites. Here are two:

site:	`ftp.nisc.sri.com`
directory:	`rfc`
file:	`rfc1341.txt`
file:	`rfc1341.ps`
mode:	`ascii`

site:	`wuarchive.wustl.edu`
directory:	`doc/rfc`
file:	`rfc1341.txt`
file:	`rfc1341.ps`
mode:	`ascii`

Here is an example of how to obtain the RFC via anonymous FTP from one site. Your input is given in *slanted* style, and responses are in `typewriter` style.

```
% ftp ftp.nisc.sri.com
Connected to phoebus.NISC.SRI.COM.
220 phoebus FTP server (SRI Version 1.98 Fri Apr 19 11:57:54 PDT
1991) ready.
Name (ftp.nisc.sri.com:jsweet):  anonymous
331 Guest login ok, send ident as password.
Password:  put your e-mail address here, e.g. jsweet@irvine.com

230 Guest login ok, access restrictions apply.
ftp> cd rfc
250 CWD command successful.
ftp> get rfc1341.txt
200 PORT command successful.
150 Opening ASCII mode data connection for rfc1341.txt (211117
bytes).
226 Transfer complete.
```

```
216382 bytes received in 36 seconds (5.8 Kbytes/s)
ftp> quit
221 Be Excellent to one another!
```

There is also a PostScript version of RFC 1341 in the same place; get `rfc1341.ps`.

RFC 1341 is also available via mail server from `nisc.sri.com`. To obtain it, send a message like this one:

```
To: mail-server@nisc.sri.com
Subject: send rfc1341.txt

send rfc1341.txt
```

▷ **Documents in the MH source tree** include these:

1. `papers/tutorial/` ("The Rand MH Message Handling System: Tutorial")
2. `papers/realwork/` ("How to process 200 messages a day and still get some real work done")
3. `miscellany/multi-media/READ-ME` (MIME program setup information)

The first two documents are in a somewhat less-well-known TeX format, so you may prefer to obtain the pre-created versions via anonymous FTP:

```
site:       ftp.ics.uci.edu
directory:  pub/mh/doc
file:       tutorial.txt.Z
file:       tutorial.ps.Z
file:       realwork.txt.Z
file:       realwork.ps.Z
mode:       binary
```

Here is an abbreviated example of how to obtain the papers and also the MH source tree via anonymous FTP. A more complete FTP example was shown previously.

```
% ftp ftp.ics.uci.edu
Name (ftp.ics.uci.edu.edu:jsweet):  anonymous
Password:  put your e-mail address here, e.g. jsweet@irvine.com

ftp> binary
ftp> cd /pub/mh/doc
ftp> get tutorial.txt.Z          —same as papers/tutorial/ above
ftp> get realwork.txt.Z          —same as papers/realwork/ above
ftp> cd /pub/mh
ftp> get mh-6.8.tar.Z
ftp> quit
```

The papers are also available in PostScript format in the same directory; get `tutorial.ps.Z` and `realwork.ps.Z`.

Other FTP mirror sites, such as `ftp.uu.net`, may also have the MH sources available for anonymous FTP.

Files whose names end with `.Z` are compressed. To read them, use the `uncompress`, `compress -d`, or `zcat` commands.

Files whose names end with `.tar.Z` are compressed files in `tar` format. To find out what's in them, use a command like this one:

```
% compress -cd mh-6.8.tar.Z | tar tvf -
```

To extract their contents, use a command like this one:

```
% compress -cd mh-6.8.tar.Z | tar xvf -
```

Note that the vendor-supplied `tar` command on some System V systems may require options in addition to `xvf` in order to permit sub-directories to be extracted properly when `tar` is being run by users other than `root`.

You can also get the MH sources on a 6250 BPI 9-track magtape, along with a printed MH documentation set (not separately available, except via FTP) by following these directions:

> Send $75 US to the address below. This covers the cost of a 6250 BPI 9-track magtape, handling, and shipping. Be sure to include your USPS address with your check. Checks must be drawn on U.S. funds and should be made payable to "Regents of the University of California."
>
> The distribution address is:

University of California at Irvine
Office of Academic Computing
360 Computer Science
Irvine, CA 92717 USA

telephone: +1 714 856 5153

- **mhn(1)** — This is the on-line manual page for the `mhn` program. It's part of the MH software installation. You should be able to read it by typing the command `man mhn`.

- *The Simple Times* can be obtained in any of its various MIME versions. To get the most up-to-date information about how to subscribe, send an e-mail message like this one:

  To: st-subscriptions@simple-times.org
  Subject: help

- **USENET news groups** contain useful information about MIME and MH. Articles in MIME format are already appearing. Two relevant news groups are `comp.mail.mime` and `comp.mail.mh`.

  There is also an Internet discussion mailing list related to MH, to which you can subscribe by requesting information from this e-mail address:

  `mh-users-request@ics.uci.edu`

- **Image and video files** are available from many sources. For weather maps, use anonymous FTP to site `vmd.cso.uiuc.edu` and look in the directory `wx`. Many other sites offer image files, such as `ftp.uu.net`, in these directories (from a brief survey):

  ```
  doc/literary/obi/Soviet.Archives/images
  doc/music/guitar/Led_Zeppelin
  networking/mail/metamail
  systems/ibmpc/msdos/simtel20/gif
  ```

  Site `ames.arc.nasa.gov` has a number of GIF-format space image files available for anonymous FTP, such as `toutatis.gif`, four views of asteroid 4179 Toutatis. The files can be found in the directory `pub/SPACE/GIF`.

  Use the FTP `binary` or `type binary` command when obtaining GIF files.

  Alternatively, simply send a message like this one to yourself:

```
To: your-address-here
Subject: 4179 Toutatis
-------
#@text/plain [4179 Toutatis info] \
        access-type=anon-ftp; \
        name="toutatis.txt"; \
        directory="pub/SPACE/GIF"; \
        site="ames.arc.nasa.gov"
#@image/gif [4179 Toutatis photos] \
        access-type=anon-ftp; \
        name="toutatis.gif"; \
        directory="pub/SPACE/GIF"; \
        site="ames.arc.nasa.gov"

^D

What now? edit mhn

What now? send
```

Some MPEG video files are available for anonymous FTP as follows:

site:	`toe.cs.berkeley.edu`
directory:	`pub/multimedia/mpeg/movies`
files:	`*.mpg`
mode:	`binary`

GIF, JPEG, and MPEG pictures also appear in some USENET news groups:

```
alt.binaries.pictures
alt.binaries.pictures.erotica
alt.binaries.pictures.fine-art.digitized
alt.binaries.pictures.fine-art.graphics
alt.binaries.pictures.fractals
alt.binaries.pictures.misc
alt.binaries.pictures.tasteless
```

Some sites do not have these groups available.

▷ **Audio files** are available on SPARCstations running SunOS 4.x in the directory `/usr/demo/SOUND/sounds/` (if installed). HP 9000/710 workstations running HP-UX 8.x do not appear to include pre-recorded sound files.

Various sound files for different kinds of workstations are often made available for anonymous FTP.

Sound files and discussions about them are posted to these USENET news groups:

```
alt.binaries.sounds.misc
alt.binaries.sounds.d
```

Be aware that not all sound files are compatible with the MIME `audio/basic` format, and may require manual conversion.

The Audio File Formats FAQ (frequently asked question and answer list), maintained by Guido van Rossum, `<guido@cwi.nl>` is posted periodically in the USENET news group `news.answers`.

▷ **Display Programs** are available from various sites on the Internet. Some experience may be required to compile the programs.

The `miscellany/multi-media/READ-ME` document in the MH source tree contains information about how to compile some of the simpler programs. The more complicated programs come with their own documentation.

`gs` —This program (aka GhostScript) is a PostScript interpreter, capable of displaying PostScript documents on X Window System displays, such as those that occur in `application/postscript` message parts. You might also want to obtain `ghostview`, an application specifically designed for viewing documents with GhostScript. `Gs` and `ghostview` are available via anonymous FTP from a number of sites; here is one:

site:	`ftp.uu.net`
directory:	`systems/gnu/ghostscript-2.5.2.tar.Z-split`
files:	`part*`
mode:	`binary`
directory:	`systems/gnu`
files:	`ghostview-1.3.tar.Z`
mode:	`binary`

`richtext` —This program displays `text/richtext` message parts using whatever simple display attributes are available for your terminal. It's part

of the MetaMail software, but it can be compiled and used separately with MH. MetaMail is available via anonymous FTP from a number of sites; here are two:

site: `ftp.uu.net`
directory: `networking/mail/metamail`
file: `mm.tar.Z`
mode: `binary`

site: `thumper.bellcore.com`
directory: `pub/nsb`
file: `mm.tar.Z`
mode: `binary`

`rt2raw` —This is available in the MH source tree. It can be used to display message parts of type `text/richtext`. It works by stripping richtext formatting commands, leaving plain ASCII text. Look for it in this directory:

```
miscellany/multi-media/misc/
```

It's also available as a very short (1 page) C source program in RFC 1341.

`mail.weather.ext` —This perl program uses `mhn` to mail weather images as external body parts. It's described in section 4.1. It's available via anonymous FTP as follows:

site: `ftp.ics.uci.edu`
directory: `mh/contrib/multi-media`
file: `mhweather.shar`
mode: `ascii`

`mpeg_play` —This program plays MPEG video files (as in `video/mpeg`) on X Window System displays. It's available via anonymous FTP as follows:

site: `toe.cs.berkeley.edu`
directory: `pub/multimedia/mpeg`
file: `mpeg_play-2.0.tar.Z`
mode: `binary`

`rec.sparc` —This perl program can be used to make simple recordings on a SPARCstation. It prompts for a Return to start the recording and then for another Return to end the recording. The `audiotool` program is better, but you can get by with `rec.sparc`. The program is available via anonymous FTP as follows:

site:	`ftp.ics.uci.edu`
directory:	`mh/contrib/multi-media`
file:	`rec.sparc`
mode:	`ascii`

`xv` —This program displays many different kinds of image formats on X Window System displays, including GIF (as in `image/gif`) and JPEG (as in `image/jpeg`). `Xv` is available via anonymous FTP from a number of sites; here are two:

site:	`ftp.cis.upenn.edu`
directory:	`pub`
files:	`xv-2.21.tar.Z`
mode:	`binary`

site:	`ftp.uu.net`
directory:	`pub/window-sys/X/contrib/xv-2.21.tar.Z-split`
files:	`part*`
mode:	`binary`

F Acknowledgements

Thanks go to Marshall T. Rose of Dover Beach Consulting for contributing *Simple Times* material to this tutorial, for making comments about this tutorial, for answering my questions, and for producing the MIME extensions to MH in the first place!

Three persons in the Support Group of the Department of Information and Computer Science at the University of California, Irvine, were very helpful:

John Romine and David Harnick-Shapiro shared their thoughts on MH and MIME with me, and provided comments about this tutorial; Tim Morgan provided pointers to `mpeg_play`.

A `getweather` shell script, by a gentleman named John Bradley, provided the inspiration for the weather map mailer program.

Irvine Compiler Corporation provided the computer resources for this tutorial.

G References

Borenstein, N. and Freed, N., "MIME (Multipurpose Internet Mail Extensions)," June 1992, Bellcore and Innosoft, RFC 1341.

Garfinkel, S. and Spafford, G., *Practical UNIX Security*, June, 1991, O'Reilly & Associates, Inc., ISBN 0-937175-72-2.

Krol, Ed, *The Whole Internet User's Guide & Catalog*, 1992, O'Reilly & Associates, Inc., ISBN 1-56592-025-2.

Peek, Jerry, *MH & xmh*, 1991, O'Reilly & Associates, Inc., ISBN 0-937175-64-1.

Rose, Marshall T. and Romine, John L., *The Rand MH Message Handling System: User's Manual, 6.8 #1*, December 1992, University of California, Irvine.

Rose, Marshall T., *The Internet Message*, 1993, P T R Prentice-Hall, Inc., ch. 6., ISBN 0-13-092941-7.

van Rossum, Guido, "FAQ: Audio File Formats (version 2.3)," July, 1992, USENET `news.answers`.

Vielmetti, Edward, "Frequently Asked Questions about MIME," February, 1993, USENET `comp.mail.mime`.