discard this page

The RAND *MH* Message Handling System: Administrator's Guide

UCI Version

November 30, 1993
6.8.3 #1[UCI]

1. INTRODUCTION

Scope of this document

This is the Administrator's Guide to *MH*. If you don't maintain an *MH* system, don't read this; the information is entirely too technical. If you are a maintainer, then read this guide until you understand it, follow the advice it gives, and then forget about the guide.

Before continuing, I'll point out two facts:

This document will never contain all the information you need to maintain MH.

Furthermore, this document will never contain everything I know about maintaining MH.

MH, and mailsystems in general, are more complex than most people realize. A combination of experience, intuition, and tenacity is required to maintain *MH* properly. This document can provide only guidelines for bringing up an *MH* system and maintaining it. There is a sufficient amount of customization possible that not all events or problems can be forseen.

Summary

During *MH* generation, you specify several configuration constants to the *mhconfig* program. These directives take into consideration such issues as hardware and operating system dependencies in the source code. They also factor out some major mailsystem administrative decisions that are likely to be made consistantly at sites with more than one host. The manual entry *mh–gen* (8) describes all the static configuration directives.

However, when you install *MH* you may wish to make some site–specific or host–specific changes which aren't hardware or even software related. Rather, they are administrative decisions. That's what this guide is for: it describes all of the dynamically tailorable directives.

Usually, after installing *MH*, you'll want to edit the **/usr/local/lib/mh/mtstailor** file. This file fine-tunes the way *MH* interacts with the message transport system (MTS). Section 2 talks about the MTS interface and MTS tailoring.

After that, if you're running the UCI BBoards facility, or the POP facility, you'll need to know how to maintain those systems. Sections 3 and 4 talk about these.

If for some reason you're not running an MTS that can handle both Internet and *UUCP* traffic, you should read–up on mail filtering in Section 5. Although this is considered ‘‘old technology’’ now, the mechanisms described in Section 5 were really quite useful when first introduced way back in 1981.

Finally, you may want to know how to modify the *MH* source tree. Section 6 talks (a little bit) about that.

The last two sections describe a few hidden features in *MH*, and the configuration options that were in effect when this guide was generated.

After *MH* is installed, you should define the address ‘‘Bug–MH’’ to map to either you or the *PostMaster* at your site.

In addition, if you want to tailor the behavior of *MH* for new users, you can create and edit the file **/usr/local/lib/mh/mh.profile**. When the *install-mh* program is run for a user, if this file exists, it will copy it into the user’s .mh_profile file.

2. THE MTS INTERFACE

The file **/usr/local/lib/mh/mtstailor** customizes certain host–specific parameters of *MH* related primarily to interactions with the transport system. The parameters in this file override the compiled–in defaults given during *MH* configuration. Rather than recompiling *MH* on each host to make minor customizations, it is easier simply to modify the **mtstailor** file. All hosts at a given site normally use the same **mtstailor** file, though this need not be the case.

It is a good idea to run the *conflict* (8) program each morning under *cron*. The following line usually suffices:

```
00 05 * * * /usr/local/lib/mh/conflict -mail PostMaster
```

NAME

mh-tailor, mtstailor – system customization for MH message handler

SYNOPSIS

/usr/local/lib/mh/mtstailor

DESCRIPTION

The file /usr/local/lib/mh/mtstailor defines run-time options for those *MH* programs which interact (in some form) with the message transport system. At present, these (user) programs are: *ap*, *conflict*, *inc*, *msgchk*, *msh*, *post*, *rcvdist*, and *rcvpack*.

Each option should be given on a single line. Blank lines and lines which begin with '#' are ignored. The options available along with default values and a description of their meanings are listed below:

localname:
: The host name *MH* considers local. If not set, depending on the version of UNIX you're running, *MH* will query the system for this value (e.g., <whoami.h>, gethostname, etc.). This has no equivalent in the *MH* configuration file. POP client hosts should set this value to the name of the POP service host.

localdomain:
: If this is set, a '.' followed by this string will be appended to your host name. This might be useful for sites where the host name returned by the system (e.g., <whoami.h>, gethostname, etc.), is not a ''fully qualified domain name'' (i.e., does not contain a '.').

clientname:
: The host name *MH* will give in the SMTP **HELO** (and **EHLO**) command, when posting mail. If not set, no **HELO** command will be given. Although the **HELO** command is required by RFC 821, many SMTP servers do not require it.

: Early versions of SendMail will fail if the host name given in the **HELO** command is the local host; later versions of SendMail will complain if you omit the **HELO** command. If you run SendMail, find out what your system expects and set this field accordingly.

systemname:
: The name of the local host in the *UUCP* ''domain''. If not set, depending on the version of UNIX you're running, *MH* will query the system for this value. This has no equivalent in the *MH* configuration file.

mmdfldir: /usr/spool/mail
: The directory where maildrops are kept. If this is empty, the user's home directory is used. This overrides the ''mail'' field in the *MH* configuration file.

mmdflfil:
: The name of the maildrop file in the directory where maildrops are kept. If this is empty, the user's login name is used. This overrides the ''mail'' field in the *MH* configuration file.

mmdelim1: \001\001\001\001\n
: The beginning-of-message delimiter for maildrops.

mmdelim2: \001\001\001\001\n

The end-of-message delimiter for maildrops.

mmailid: 0

If non-zero, then support for MMailids in **/etc/passwd** is enabled. Basically, the pw_gecos field in the password file is of the form

My Full Name <mailid>

The *MH* internal routines that deal with user and full names will return ‘‘mailid’’ and ‘‘My Full Name’’ respectively.

lockstyle: 0

The locking discipline to perform. A value of ‘‘0’’ means to use kernel-level locking if available. (See below for more details.) On systems compiled without kernel-level locking, standard *BellMail* locking is used. A value of ‘‘1’’ means to use *BellMail* locking always (the name of the lock is based on the file name). A value of ‘‘2’’ means to use *MMDF* locking always (the name of the lock is based on device/inode pairs).

lockldir:

The name of the directory for making locks. If your system isn’t configured to use kernel-level locking, then this directory is used when creating locks. If the value is empty, then the directory of the file to be locked is used.

maildelivery: /usr/local/lib/mh/maildelivery

The name of the system-wide default *.maildelivery* file. See *mhook* (1) for the details.

everyone: 200

The highest user-id which should NOT receive mail addressed to ‘‘everyone’’.

noshell:

If set, then each user-id greater than ‘‘everyone’’ that has a login shell equivalent to the given value (e.g., ‘‘/bin/csh’’) indicates that mail for ‘‘everyone’’ should not be sent to them. This is useful for handling admin, dummy, and guest logins.

Mail Filtering

These options are only available if you compiled *MH* with ‘‘options MF’’.

uucpchan: name of *UUCP* channel

Usually ‘‘UUCP’’. This has no equivalent in the *MH* configuration file.

uucpldir: /usr/spool/mail

The name of the directory where *UUCP* maildrops are kept. This has no equivalent in the *MH* configuration file.

uucplfil:

The name of the maildrop file in the directory where *UUCP* maildrops are kept. If this is empty, the user’s login name is used. This has no equivalent in the *MH* configuration file.

umincproc: /usr/local/lib/mh/uminc

The path to the program that filters *UUCP*-style maildrops to *MMDF*-style maildrops.

Stand-Alone Delivery

These options are only available if you compiled *MH* to use stand-alone delivery (i.e., ‘‘mts: mh’’).

mailqdir: /usr/spool/netmail
: The directory where network mail is queued.

tmailqdir: /usr/tmp
: The directory where network mail queue files are built.

syscpy: 1
: If ON, unauthorized mail is copied to the overseer.

overseer: root
: The user that receives reports of unauthorized mail.

mailer: root
: The user acting for the mail system.

fromtmp: /tmp/rml.f.XXXXXX
: The *mktemp* template for storing from lines.

msgtmp: /tmp/rml.m.XXXXXX
: The *mktemp* template for storing the rest of the message.

errtmp: /tmp/rml.e.XXXXXX
: The *mktemp* template for storing error messages from other mailers.

tmpmode: 0600
: The octal mode which temporary files are set to.

okhosts: /usr/local/lib/mh/Rmail.OKHosts
: A file containing a list of hosts that can send ARPAnet mail.

okdests: /usr/local/lib/mh/RMail.OKDests
: A file containing a list of hosts that can always receive mail.

The ‘/smtp’ MTS Suffix

These options are only available if you compiled *MH* with the ‘‘/smtp’’ suffix to your ‘‘mts:’’ configuration.

hostable: /usr/local/lib/mh/hosts
: The exceptions file for /etc/hosts used by *post* to try to find official names. The format of this file is quite simple:

1. Comments are surrounded by sharp (‘#’) and newline.
2. Words are surrounded by white space.
3. The first word on the line is the official name of a host.
4. All words following the official names are aliases for that host.

servers: localhost \01localnet

A lists of hosts and networks which to look for SMTP servers when posting local mail. It turns out this is a major win for hosts which don't run an message transport system. The value of ‘‘servers’’ should be one or more items. Each item is the name of either a host or a net (in the latter case, precede the name of the net by a \01). This list is searched when looking for a smtp server to post mail. If a host is present, the SMTP port on that host is tried. If a net is present, the SMTP port on each host in that net is tried. Note that if you are running with the BIND code, then any networks specified are ignored (sorry, the interface went away under BIND).

SendMail

This option is only available if you compiled *MH* to use *SendMail* as your delivery agent (i.e., ‘‘mts: sendmail’’).

sendmail: /usr/lib/sendmail
The pathname to the *sendmail* program.

Post Office Protocol

This option is only available if you compiled *MH* with POP support enabled (i.e., ‘‘pop: on’’).

pophost:
The name of the default POP service host. If this is not set, then *MH* looks in the standard maildrop areas for waiting mail, otherwise the named POP service host is consulted.

BBoards Delivery

This option is only available if you compiled *MH* with ‘‘bbdelivery: on’’.

bbdomain:
The local BBoards domain (a UCI hack).

BBoards & The POP

These options are only available if you compiled *MH* with ‘‘bboards: pop’’ and ‘‘pop: on’’.

popbbhost:
The POP service host which also acts as a BBoard server. This variable should be set on the POP BBoards client host.

popbbuser:
The guest account on the POP/BB service host. This should be a different login ID than either the POP user or the BBoards user. (The user-id ‘‘ftp’’ is highly recommended.) This variable should be set on both the POP BBoards client and service hosts.

popbblist: /usr/local/lib/mh/hosts.popbb
A file containing of lists of hosts that are allowed to use the POP facility to access BBoards using the guest account. If this file is not present, then no check is made. This variable should be set on the POP BBoards service host.

BBoards & The NNTP

This option is only available if you compiled *MH* with ‘‘bboards: nntp’’ and ‘‘pop: on’’.

nntphost:

The host which provides the NNTP service. This variable should be set on the NNTP BBoards client host.

File Locking

A few words on locking: *MH* has a flexible locking system for making locks on files. There are two **mtstailor** variables you should be aware of ‘‘lockstyle’’ and ‘‘lockldir’’. The first controls the method of locking, the second says where lock files should be created.

The ‘‘lockstyle’’ variable can take on three values: 0, 1, 2. A value of 0 is useful on systems with kernel-level locking. If you are on a **BSD42** system, *MH* assumes you have the *flock* system call. On other systems: define **FLOCK** if you want to use the *flock* system call; define **LOCKF** if you want to use the *lockf* system call; or define **FCNTL** if you want to use the *fcntl* system call for kernel-level locking. If you haven’t configured *MH* to use kernel-level locking, a locking style of 0 is considered the same as locking style 1.

A value of 1 or 2 specifies that a file should be created whose existence means ‘‘locked’’ and whose non-existence means ‘‘unlocked’’. A value of 1 says to construct the lockname by appending ‘‘.lock’’ to the name of the file being locked. A value of 2 says to construct the lockname by looking at the device and inode numbers of the file being locked. If the ‘‘lockldir’’ variable is not specified, lock files will be created in the directory where the file being locked resides. Otherwise, lock files will be created in the directory specified by ‘‘lockldir’’. Prior to installing *MH*, you should see how locking is done at your site, and set the appropriate values.

Files

/usr/local/lib/mh/mtstailor	tailor file

Profile Components

None

See Also

mh–gen(8), mh–mts(8)

Defaults

As listed above

Context

None

NAME

mh-mts – the MH interface to the message transport system

SYNOPSIS

SendMail

Zmailer

MMDF (any release)

stand–alone

DESCRIPTION

MH can use a wide range of message transport systems to deliver mail. Although the *MH* administrator usually doesn't get to choose which MTS to use (since it's already in place), this document briefly describes the interfaces.

When communicating with *SendMail*, *MH* always uses the SMTP to post mail. Depending on the *MH* configuration, *SendMail* may be invoked directly (via a *fork* and an *exec*), or *MH* may open a TCP/IP connection to the SMTP server on the localhost.

When communicating with *zmailer*, the *SendMail* compatibility program is required to be installed in /usr/lib. *MH* communicates with *zmailer* by using the SMTP. It does this by invoking the **/usr/lib/sendmail** compatibility program directly, with the '–bs' option.

When communicating with *MMDF*, normally *MH* uses the ''mm_'' routines to post mail. However, depending on the *MH* configuration, *MH* instead may open a TCP/IP connection to the SMTP server on the localhost.

When using the stand–alone system (**NOT** recommended), *MH* delivers local mail itself and queues *UUCP* and network mail. The network mail portion will probably have to be modified to reflect the local host's tastes, since there is no well–known practice in this area for all types of UNIX hosts.

If you are running a UNIX system with TCP/IP networking, then it is felt that the best interface is achieved by using either *SendMail* or *MMDF* with the SMTP option. This gives greater flexibility. To enable this option you append the /smtp suffix to the mts option in the *MH* configuration. This yields two primary advantages: First, you don't have to know where *submit* or *SendMail* live. This means that *MH* binaries (e.g., *post*) don't have to have this information hard–coded, or can run different programs altogether; and, second, you can post mail with the server on different systems, so you don't need either *MMDF* or *SendMail* on your local host. Big win in conserving cycles and disk space. Since *MH* supports the notion of a server search–list in this respect, this approach can be tolerant of faults. Be sure to set ''servers:'' as described in mh–tailor(8) if you use this option.

There are four disadvantages to using the SMTP option: First, only UNIX systems with TCP/IP are supported. Second, you need to have an SMTP server running somewhere on any network your local host can reach. Third, this bypasses any authentication mechanisms in *MMDF* or *SendMail*. Fourth, the file **/etc/hosts** is used for hostname lookups (although there is an exception file). In response to these disadvantages though: First, there's got to be an SMTP server somewhere around if you're in the Internet or have a local network. Since the server search–list is very general, a wide–range of options are possible. Second, SMTP should be fixed to have authentication mechanisms in it, like POP. Third, *MH* won't choke on mail to hosts whose official names it can't verify, it'll just plug along (and besides if you enable the BERK or

DUMB configuration options, *MH* ignores the hosts file altogether).

Files

/usr/local/lib/mh/mtstailor	tailor file

Profile Components

None

See Also

MMDF–II: A Technical Review, Proceedings, Usenix Summer ’84 Conference
SENDMAIL — An Internetwork Mail Router
mh–tailor(8), post(8)

Defaults

None

Context

None

Bugs

The /usr/local/lib/mh/mtstailor file ignores the information in the *MMDF–II* tailoring file. It should not.

3. BBOARDS

The UCI BBoards facility has two aspects: message reading, and message delivery. The configuration directives applicable to BBoards are ‘‘bboards: on/off/pop/nntp’’ and ‘‘bbdelivery: on/off’’.

BBoard Delivery

If you enabled BBoards delivery (‘‘bbdelivery: on’’) during configuration, then the initial environment for bboards delivery was set–up during installation. A BBoard called ‘‘system’’ is established, which is the BBoard for general discussion.

To add more BBoards, become the ‘‘bboards’’ user, and edit the **/usr/spool/bboards/BBoards** file. The file **support/bboards/Example** is a copy of the **/usr/spool/bboards/BBoards** file that we use at UCI. When you add a BBoard, you don’t have to create the files associated with it, the BBoards delivery system will do that automatically.

Private BBoards may be created. To add the fictitious private BBoard ‘‘hacks’’, add the appropriate entry to the BBoards file, create the empty file **/usr/spool/bboards/hacks.mbox** (or whatever), change the mode of this file to 0640, and change the group of the file to be the groupid of the people that you want to be able to read it. Also be sure to add the ‘‘bboards’’ user to this group (in **/etc/group**), so the archives can be owned correctly.

By using the special INVIS flag for a BBoard, special purpose BBoards may be set–up which are invisible to the *MH* user. For example, if a site distributes a BBoard both locally to a number of machines and to a number of distant machines. It might be useful to have two distribution lists: one for all machines on the list, and the other for local machines only. This is actually very simple to do. For the main list, put the standard entry of information in the **/usr/spool/bboards/BBoards** file, with the complete distribution list. For the local machines list, and add a similar entry to the **/usr/spool/bboards/BBoards** file. All the fields should be the same except three: the BBoard name should reflect a local designation (e.g., ‘‘l–hacks’’), the distribution list should contain only machines at the local site, and the flags field should contain the INVIS flag. Since the two entries share the same primary and archive files, messages sent to either list are read by local users, while only thoses messages sent to the main list are read by all users.

Two automatic facilities for dealing with BBoards exist: automatic archiving and automatic aliasing. The file **support/bboards/crontab** contains some entries that you should add to your **/usr/lib/crontab** file to run the specified programs at times that are convenient for you. The **bboards.daily** file is run once a day and generates an alias file for *MH*. By using this file, users of *MH* can use, for example, ‘‘unix–wizards’’ instead of ‘‘unix–wizards@brl–vgr’’ when they want to send a message to the ‘‘unix–wizards’’ discussion group. This is a major win, since you just have to know the name of the group, not the address where it’s located.

The **bboards.weekly** file is run once a week and handles old messages (those received more than 12 days ago) in the BBoards area. In short, those BBoards which are marked for automatic archiving will have their old messages placed in the **/usr/spool/bboards/archive/** area, or have their old messages removed. Not only does this make BBoards faster to read, but it conveniently partitions the new messages from the old messages so you can easily put the old messages on tape and then remove them. It turns out that this automatic archiving capability is also a major win.

At UCI, our policy is to save archived messages on tape (every two months or so). We use a program called *bbtar* to implement our particular policy. Since some BBoards are private (see above), we save the archives on two tapes: one containing the world–readable archives (this tape is read-only accessible to all users by calling the operator), and the other containing the non–world–readable ones (this tape is kept locked–up somewhere).

BBoards with the POP

If you configured *MH* with ‘‘bboards: pop’’ and ‘‘pop: on’’, then the *MH* user is allowed to read BBoards on a server machine instead of the local host (thus saving disk space). For completely transparent behavior, the administrator may set certain variables in the **mtstailor** file on the client host. The variable ‘‘popbbhost’’ indicates the host where BBoards are kept (it doesn’t have to be the POP service host, but this host must run both a POP server and the BBoards system). The variable ‘‘popbbuser’’ indicates the guest account on this host for BBoards. This username should not be either the POP user or the BBoards user. Usually the anonymous FTP user (ftp) is the best choice. Finally, the variable ‘‘popbblist’’ indicates the name of a file which contains a list of hosts (one to a line, official host names only) which should be allowed to use the POP facility to access BBoards via the guest account. (If the file is not present, then no check is made.)

The ‘‘popbbuser’’ variable should be set on both the client and service host. The ‘‘popbbhost’’ variable need be set only on the client host (the value, of course, is the name of the service host). The ‘‘popbblist’’ variable need be set only on the service host.

Finally, on the client host, if a POP service host is not explicitly given by the user (i.e., ‘‘popbbhost’’ is implicitly used), then *bbc* will explicitly check the local host prior to contacting the service host. This allows each POP client host to have a few local BBoards (e.g., each host could have one called ‘‘system’’), and then have the POP service host used for all the rest (a site–wide BBoard might be known as ‘‘general’’).

BBoards with the NNTP

If you configured *MH* with ‘‘bboards: nntp’’ and ‘‘pop: on’’, then the *MH* user is allowed to read the Network News on a server machine using the standard *bbc* command. For completely transparent behavior, the administrator may set the ‘‘nntphost’’ variable in the **mtstailor** file to indicate the host where the Network News is kept. The ‘‘nntphost’’ variable should be set only on the client host Finally, on the client host, if an NNTP service host is not explicitly given by the user (i.e., ‘‘nntphost’’ is implicitly used), then *bbc* will explicitly check the local host prior to contacting the service host. This allows each NNTP client host to have a few local BBoards (e.g., each host could have one called ‘‘system’’), and then have the NNTP service host used for to read the Network News.

Reading BBoards via the POP and via the NNTP are mutually exclusive.

NAME

BBoards – BBoards database

SYNOPSIS

/usr/spool/bboards/BBoards

DESCRIPTION

The BBoards database contains for each BBoard the following information:

field	*value*
name	the name of the BBoard
aliases	local aliases for the BBoard (separated by commas)
primary file	the .mbox file
encrypted password	leadership password
leaders	local list maintainers (separated by commas) usernames from the *passwd* (5) file, or groupnames preceded by '=' from the *group* (5) file
network address	the list address
request address	the list maintainer's address
relay	the host acting as relay for the local domain
distribution sites	(separated by commas)
flags	special flags (see <bboards.h>)

This is an ASCII file. Each field within each BBoard's entry is separated from the next by a colon. Each BBoard entry is separated from the next by a new-line. If the password field is null, no password is demanded; if it contains a single asterisk, then no password is valid.

This file resides in the home directory of the login ‘‘bboards’’. Because of the encrypted passwords, it can and does have general read permission.

Files

/usr/spool/bboards/BBoards	BBoards database

See Also

bbaka(8), bbexp(8), bboards (8), bbtar(8)

Bugs

A binary indexed file format should be available for fast access.

Appropriate precautions must be taken to lock the file against changes if it is to be edited with a text editor. A *vibb* program is needed.

NAME

bbaka – generate an alias list for BBoards

SYNOPSIS

/usr/spool/bboards/bbaka [system]

DESCRIPTION

The *bbaka* program reads the BBoards database and produces on its standard output a file suitable for inclusion in either the *MMDF–II* aliases file (if the argument ‘system’ is given). If the argument is not given, then *bbaka* produces on its standard output a file suitable for becoming the /usr/local/lib/mh/BBoardsAliases file.

Files

/usr/spool/bboards/BBoards	BBoards database
/usr/local/lib/mh/BBoardsAliases	BBoards aliases file for *MH*

Profile Components

None

See Also

bboards(5)

Defaults

None

Context

None

NAME

bbexp – expunge the BBoards area

SYNOPSIS

/usr/spool/bboards/bbexp [–*first–metric*] [–*second–metric*] [bboards ...]

DESCRIPTION

The *bbexp* program reads the BBoards database and calls *msh* to archive the named BBoards (or all BBoards if none are specified).

The first–metric (which defaults to 12) gives the age in days of the ‘‘BB–Posted:’’ field for messages which should be expunged. The second–metric (which defaults to 20) gives the age in days of the ‘‘Date:’’ field for messages which should be expunged. Any message which meets either metric will be either archived or removed, depending on what the *BBoards* (5) file says.

Files

/usr/spool/bboards/BBoards	BBoards database

Profile Components

None

See Also

msh(1), bboards(5)

Defaults

None

Context

None

NAME

bboards – BBoards channel/mailer

SYNOPSIS

/usr/mmdf/chans/bboards fd1 fd2 [y]

/usr/local/lib/mh/sbboards bboard ...

/usr/local/lib/mh/sbboards file maildrop directory bboards.bboard

DESCRIPTION

For *MMDF*, the BBoards channel delivers mail to the BBoards system. For *SendMail* and stand–alone *MH*, the SBBoards mailer performs this task.

For each address given, these programs consult the *bboards* (5) file to ascertain information about the BBoard named by the address. The programs then perform local delivery, if appropriate. After that, with the exception of *sbboards* running under stand–alone *MH*, the programs perform redistribution, if appropriate.

For redistribution, the return address is set to be the request address at the local host, so bad addresses down the line return to the nearest point of authority. If any failures occur during redistribution, a mail message is sent to the local request address.

Files

/usr/local/lib/mh/mtstailor	tailor file
/usr/spool/bboards/BBoards	BBoards database

Profile Components

None

See Also

bboards(5), bbaka(8)

Defaults

None

Context

None

NAME

bbtar – generate the names of archive files to be put to tape

SYNOPSIS

/usr/spool/bboards/bbtar [private] [public]

DESCRIPTION

The *bbtar* program reads the BBoards database and produces on its standard output the names of BBoards archives which should be put to tape, for direct use in a *tar* (1) command.

If the argument ‘private’ is given, only private BBoards are considered. If the argument ‘public’ is given, only public BBoards are considered. This lets the BBoards administrator write two tapes, one for general read–access (the public BBoards), and one for restricted access. The default is all BBoards

For example:

```
cd archive              # change to the archive directory
tar cv ‘bbtar private‘  # save all private BBoard archives
```

After the archives have been saved to tape, they are usually removed. The archives are then filled again, usually automatically by cron jobs which run *bbexp* (8).

Files

/usr/spool/bboards/BBoards	BBoards database

Profile Components

None

See Also

bboards(5), bbexp(8)

Defaults

None

Context

None

4. POP

For POP (Post Office Protocol) client hosts, you need to edit the **/usr/local/lib/mh/mtstailor** file to know about two hosts: the SMTP service host and the POP service host. Normally, these are the same. Change the ‘‘localname’’ field of the **mtstailor** file of *MH* in the file to be the name of the POP service host. This makes replies to mail generated on the POP client host possible, since *MH* will consider use the hostname of the POP service host as the local hostname for outgoing mail. Also set the value of ‘‘pophost’’ to this value. This tells *inc* and *msgchk* to use POP instead of looking for mail locally. Finally, make sure the value of ‘‘servers’’ includes the name of the SMTP service host. The recommended value for ‘‘servers’’ is:

```
servers: SMTP–service–host localhost \01localnet
```

If you want more information on the Post Office Protocol used by *MH*, consult the files **support/pop/rfc1081.txt** and **support/pop/rfc1082.txt** which describe the *MH* version of the POP: POP3.

For POP service hosts, you need to run a daemon, *popd* (8). The daemon should start at multi–user boot time, so adding the lines:

```
if [ –f /etc/popd ]; then
   /etc/popd & echo –n ' pop'                    >/dev/console
fi
```

to the **/etc/rc.local** file is sufficient.

The port assigned to the POP3 protocol is ‘‘110’’. For historical reasons, many sites are using port ‘‘109’’ which is the port assigned to the ‘‘POP’’ (version 1 and 2) protocol. The configuration option ‘‘POPSERVICE’’ is the name of the port number that *MH* POP will try to use, and defaults to the name ‘‘pop’’.

To generate *MH* to use newer assigned port number, in your *MH* config file, add:

```
options   POPSERVICE='‘‘pop3’’'
```

And on both the POP client and service hosts, you need to define the port that the POP service uses. Add the line:

```
pop3              110/tcp
```

to the **/etc/services** file (if it’s not already there).

There are two ways to administer POP: In ‘‘naive’’ mode, each user-id in the *passwd* (5) file is considered a POP subscriber. No changes are required for the mailsystem on the POP service host. However, this method requires that each POP subscriber have an entry in the password file. The POP server will fetch the user’s mail from wherever maildrops are kept on the POP service host. This means that if maildrops are kept in the user’s home directory, then each POP subscriber must have a home directory.

In ‘‘smart’’ mode (enabled via ‘‘DPOP’’ being given as a configuration option), the list of POP subscribers and the list of login users are completely separate name spaces. A separate database (simple file similar to the *BBoards* (5) file) is used to record information about each

POP subscriber. Unfortunately, the local mailsystem must be changed to reflect this. This requires two changes (both of which are simple): First, the aliasing mechanism is augmented so that POP subscriber addresses are diverted to a special delivery mechanism. *MH* comes with a program, *popaka* (8), which generates the additional information to be put in the mailsystem's alias file. Second, a special POP channel (for MMDF-II) or POP mailer (for SendMail) performs the actual delivery (*mh.6* supplies both). All it really does is just place the mail in the POP spool area.

These two different philosophies are not compatible on the same POP service host: one or the other, but not both may be run. Clever mailsystem people will note that the POP mechanism is really a special case of the more general BBoards mechanism.

In addition, there is one user-visible difference, which the administrator controls the availability of. The difference is whether the POP subscriber must supply a password to the POP server: The first method uses the standard ARPA technique of sending a username and a password. The appropriate programs (*inc*, *msgchk*, and possibly *bbc*) will prompt the user for this information.

The second method (which is enabled via ‘‘RPOP’’ being given as a configuration option) uses the Berkeley UNIX reserved port method for authentication. This requires that the two or three mentioned above programs be *setuid* to root. (There are no known holes in any of these programs.)

To add a POP subscriber, for the first method, one simply follows the usual procedures for adding a new user, which eventually results in adding a line to the *passwd* (5) file; for the second method, one must edit the POP database file (kept in the home directory of the POP user), and then run the *popaka* program. The output of this program is placed in the aliases file for the transport system (e.g., **/usr/lib/aliases** for SendMail).

Authentication for POP subscribers differs depending on the two methods. When the user supplies a password for the POP session: under the first method, the contents of the password field for the user's entry in the *passwd* (5) is consulted; under the second method, the contents of the password field for the subscriber's entry in the *pop* (5) file is consulted. (To set this field, the *popwrd* (8) program is used.)

If you are allowing RPOP, under the first method, the user's *.rhosts* file is consulted; under the second method, the contents of the network address field for the subscriber's entry in the *pop* (5) file is consulted.

In addition, a third authentication scheme is available. When the APOP configuration option is given, e.g.,

```
options   APOP=’‘‘/etc/pop.auth’’’
```

In this case, the server also allows a client to supply authentication credentials to provide for origin authentication and reply protection, but which do not involve sending a password in the clear over the network. A POP authorization DB, having as its name the value of APOP configuration option, is used to keep track of this information. This file is created and manipulated by the *popauth* (8) program. Because this file contains secret information, it must be protected mode 0600 and owned by the super-user. Hence, your first step after installing the software is to issue

```
# popauth -init
```

which creates and initalizes the POP authorization DB.

NAME

POP – POP database of subscribers

SYNOPSIS

/usr/spool/pop/POP

DESCRIPTION

The POP database has exactly the same format as the *BBoards* (5) database, although many fields are unused. Currently, only four fields are examined:

field	*value*
name	the POP subscriber
primary file	the maildrop for the POP subscriber (relative to the POP directory)
encrypted password	the POP subscriber's password
network address	the remote user allowed to RPOP

This is an ASCII file. Each field within each POP subscriber's entry is separated from the next by a colon. Each POP subscriber is separated from the next by a new-line. If the password field is null, then no password is valid.

To add a new POP subscriber, edit the file adding a line such as

```
mrose::mrose::::::0
```

Then, use *popwrd* to set the password for the POP subscriber. If you wish to allow POP subscribers to access their maildrops without supplying a password (by using privileged ports), fill–in the network address field, as in:

```
mrose::mrose:::mrose@nrtc-isc::::0
```

which permits ‘‘mrose@nrtc–isc’’ to access the maildrop for the POP subscriber ‘‘mrose’’. Multiple network addresses may be specified by separating them with commas, as in:

```
dave::dave:9X5/m4yWHvhCc::dave@romano.wisc.edu,dave@rsch.wisc.edu::::
```

To disable a POP subscriber from *receiving* mail, set the primary file name to the empty string. To prevent a POP subscriber from *picking–up* mail, set the encrypted password to ‘‘*’’ and set the network address to the empty string.

This file resides in home directory of the login ‘‘pop’’. Because of the encrypted passwords, it can and does have general read permission.

Files

/usr/spool/pop/POP	POP database

See Also

bboards(5), pop(8), popaka(8), popd(8), popwrd(8)

Bugs

A binary indexed file format should be available for fast access.

Appropriate precautions must be taken to lock the file against changes if it is to be edited with a text editor. A *vipop* program is needed.

NAME

pop – POP channel/mailer

SYNOPSIS

/usr/mmdf/chans/pop fd1 fd2 [y]

/usr/local/lib/mh/spop POP–subscriber ...

DESCRIPTION

For *MMDF–II*, the POP channel delivers mail to the POP spool area for later retrieval by POP subscribers. For *SendMail*, the SPOP mailer performs this task.

For each address given, these programs consult the *pop* (5) file to obtain information about the POP–subscriber named by the address. The programs then deliver the message to the spool area for the POP–subscriber.

Files

/usr/local/lib/mh/mtstailor	tailor file
/usr/spool/pop/POP	POP database

Profile Components

None

See Also

bboards(5), bbaka(8)

Defaults

None

Context

None

NAME

popaka – generate POP entries for SendMail or MMDF–II alias file

SYNOPSIS

/usr/local/lib/mh/popaka

DESCRIPTION

The *popaka* program reads the POP database and produces on its standard output a file suitable for inclusion in the SendMail or *MMDF–II* aliases file. The contents of this file divert mail for POP subscribers to the POP channel.

Files

/usr/spool/pop/POP	POP database

Profile Components

None

See Also

pop(5)

Defaults

None

Context

None

NAME

popauth - manipulate POP authorization DB

SYNOPSIS

popauth [–init] [–list] [–user name] [–help]

DESCRIPTION

The *popauth* program allows a POP-subscriber to change the secret value used to generate their authentication credentials. In addition, the super–user or master POP user may use this program to either initialize the database or to print public information from it. *popauth* is useful only when the APOP configuration option is defined. (This configuration option defines the name of the POP authorization DB.)

Under normal usage, *popauth* prompts for a new secret, just like the *passwd* program. It then updates the POP authorization DB accordingly.

With the '–init' switch, the super-user or master POP user can create a new (or zero the existing) POP authorization DB.

With the '–list' switch, the super-user or master POP user can print out public information about the named subscriber (or all subscribers).

Files

/etc/pop.auth.*	POP authorization DB

Profile Components

None

See Also

popd(8)

Defaults

None

Context

None

NAME

popd – the POP server

SYNOPSIS

/usr/etc/popd [–p portno] (under /etc/rc.local)

DESCRIPTION

The *popd* server implements the Post Office Protocol (version 3), as described in RFC1081 and RFC1082. Basically, the server listens on the TCP port named ‘‘pop’’ for connections and enters the POP upon establishing a connection. The ‘–p’ option overrides the default TCP port. If the POP2 configuration option is defined, then the server also implements version 2 of the protocol. If the APOP configuration option is defined, then the server supports a non-standard mechanism for identity-establishment in which authentication credentials are used to provide for origin authentication and reply protection, but which do not involve sending a password in the clear over the network. See *popauth*(8) for more details.

Files

/usr/spool/pop/POP	POP database

Profile Components

None

See Also

Post Office Protocol - version 3 (aka RFC–1081),
Post Office Protocol - version 3: Extended service offerings (RFC–1082),
pop(5)

Defaults

None

Context

None

History

For historical reasons, the *MH* POP defaults to using the port named ‘‘pop’’ (109) instead of its newly assigned port named ‘‘pop3’’ (110). See the POPSERVICE configuration option for more details.

Previous versions of the server (10/28/84) had the restriction that the POP client may retrieve messages for login users only. This restriction has been lifted, and true POB support is available (sending mail to a mailbox on the POP service host which does not map to a user–id in the password file).

NAME

popwrd – set password for a POP subscriber

SYNOPSIS

/usr/local/lib/mh/popwrd POP–subscriber

DESCRIPTION

The *popwrd* program lets the super–user or the master POP user or a ‘‘leader’’ of a POP subscriber change the password field for the POP subscriber in the POP database. This program is very similar to the *passwd* (1) program.

Since only the super–user and the master POP user may change any other fields of the POP database (using an ordinary editor), it is possible for the system administrator to delegate responsibility to others to manage groups of POP subscribers.

Files

/usr/spool/pop/POP	POP database

Profile Components

None

See Also

pop(5)

Defaults

None

Context

None

Bugs

Although *popwrd* does locking against other invocations of *popwrd*, editor locking for the POP database in general is not implemented. A *vipop* program is needed.

5. MAIL FILTERING

There was a time when users on a UNIX host might have had two maildrops: one from *MMDF* and the other from *UUCP*. This was really a bad problem since it prevented using a single user–interface on all of your mail. Furthermore, if you wanted to send a message to addresses on different mailsystems, you couldn't send just one message. To solve all these problems, the notion of *mail filtering* was developed that allowed sophisticated munging and relaying between the two pseudo–domains.

MH will perform mail filtering, transparently, if given the MF configuration option. However, with the advent of *SendMail* and further maturation of *MMDF*, *MH* doesn't really need to do this anymore, since these message transport agents handle it.

The mail–filtering stuff is too complicated. It should be simpler, but, protocol translation really *is* difficult.

NAME

muinc, musift, uminc, umsift – mail filters

SYNOPSIS

/usr/local/lib/mh/muinc

/usr/local/lib/mh/musift [files ...]

/usr/local/lib/mh/uminc

/usr/local/lib/mh/umsift [files ...]

DESCRIPTION

The mail filters are a set of programs that filter mail from one format to another. In particular, *UUCP*– and *MMDF*–style mail files are handled.

muinc filters mail from the user's *MMDF* maildrop into the user's *UUCP* maildrop; similarly, *uminc* filters mail from the user's *UUCP* maildrop into the user's *MMDF* maildrop. These two programs respect each system's maildrop locking protocols.

musift filters each file on the command line (or the standard input if no arguments are given), and places the result on the standard output in *UUCP* format. The files (or standard input) are expected to be in *MMDF* format. *umsift* does the same thing filtering *UUCP* formatted files (or input), and places the *MMDF* formatted result on the standard output. No locking protocols are used by these programs.

If the files aren't in the expected format, the mail filters will try to recover. In really bad cases, you may lose big.

Files

/usr/spool/mail/	UUCP spool area for maildrops
/usr/spool/mail/$USER	Location of standard maildrop

Profile Components

None

See Also

Proposed Standard for Message Header Munging (aka RFC–886),
inc(1)

Defaults

Context

Bugs

Numerous; protocol translation is very difficult.

NAME

rmail – UUCP interface to mail

SYNOPSIS

rmail address ...

DESCRIPTION

Rmail is intended as a replacement for those systems without *SendMail* or *MMDF*. It is normally invoked by *uux* on behalf of the remote *UUCP* site. For each address, it decides where to send it: either locally, via another *UUCP* link, or via the Internet.

Rmail implements a crude access control facility by consulting the files **Rmail.OkHosts** and **Rmail.OkDests** in the **/usr/local/lib/mh/** directory. Hosts listed in the former file can send messages to anywhere they please. Hosts listed in the latter file can receive messages from anywhere. Note that a host listed in the first file is implicitly listed in the second file.

Files

/usr/local/lib/mh/mtstailor	tailor file
/usr/local/lib/mh/Rmail.OkHosts	list of privileged hosts
/usr/local/lib/mh/Rmail.OkDests	list of privileged destinations

Profile Components

None

See Also

mf(1)

Defaults

None

Context

None

6. MH HACKING

Finally, here's a little information on modifying the *MH* sources. A word of advice however:

DON'T

If you really want new *MH* capabilities, write a shell script instead. After all, that's what UNIX is all about, isn't it?

Here's the organization of the *MH* source tree.

conf/	configurator tree	
config/	compiled configuration constants	
dist/	distributor	
doc/	manual entries	
h/	include files	
miscellany/	various sundries	
mts/	MTS–specific areas	
	mh/	standalone delivery
	mmdf/	MMDF–I, MMDF–II
	sendmail/	SendMail, SMTP
papers/	papers about *MH*	
sbr/	subroutines	
support/	support programs and files	
	bboards/	UCI BBoards facility
	general/	templates
	pop/	POP facility
tma/	Trusted Mail Agent (not present in all distributions)	
uip/	programs	
zotnet/	MTS–independent areas	
	bboards/	UCI BBoards facility
	mf/	Mail Filtering
	mts/	MTS constants
	tws/	date routines

NAME

mh-hack – how to hack MH

SYNOPSIS

big hack attack

DESCRIPTION

This is a description of how one can modify the *MH* system. The *MH* distribution has a lot of complex inter–relations, so before you go modifying any code, you should read this and understand what is going on.

ADDING A NEW PROGRAM

Suppose you want to create a new *MH* command called ‘‘pickle’’. First, create and edit ‘‘pickle.c’’ in the **uip/** directory. Next edit **conf/makefiles/uip** to include ‘‘pickle’’. This file has directions at the end of it which explain how it should be modified. Next, update any documentation (described below). At this point you can re–configure *MH*. See *mh–gen(8)* for instructions on how to do this (basically, you want ‘‘mhconfig MH’’).

ADDING A NEW SUBROUTINE

Suppose you want to create a new *MH* routine called ‘‘pickle’’. First, create and edit ‘‘pickle.c’’ in the **sbr/** directory. Next edit **conf/makefiles/sbr** to include ‘‘pickle’’. This file has directions at the end of it which explain how it should be modified. You should modify **config/mh.h** to define ‘‘pickle ();’’. Similarly, **sbr/llib–lsbr** should be modified for *lint*. At this point you can re–configure *MH*.

UPDATING DOCUMENTATION

Edit whatever files you want in **conf/doc/**. When documenting a new program, such as ‘‘pickle’’, you should create a manual page with the name ‘‘pickle.rf’’. The file **conf/doc/template** has a manual page template that you can use. If you are documenting a new program, then you should also update three other files: The file **conf/doc/mh.rf** should be modified to include the ‘‘.NA’’ section from ‘‘pickle.rf’’. The file **conf/doc/mh–chart.rf** should be modified to include the ‘‘.SY’’ section from ‘‘pickle.rf’’. Finally, the file **conf/doc/MH.rf** should be modified to include a ‘‘.so pickle.me’’. Naturally, none of these changes will be reflected in the configuration until you actually run *mhconfig*.

Files

Too numerous to mention. Honest.

See Also

mh–gen(8)

Bugs

Hacking is an art, but most programmers are butchers, not artists.

7. HIDDEN FEATURES

The capabilities discussed here should not be used on a production basis, as they are either experimental, are useful for debugging *MH*, or are otherwise not recommended.

Debug Facilities

The *mark* command has a '–debug' switch which essentially prints out all the internal *MH* data structures for the folder you're looking at.

The *post* command has a '–debug' switch which does everything but actually post the message for you. Instead of posting the draft, it sends it to the standard output. Similarly, *send* has a '–debug' switch which gets passed to *post*.

Some *MH* commands look at envariables to determine debug–mode operation of certain new facilities. The current list of envariables is:

MHFDEBUG	OVERHEAD facility
MHLDEBUG	mhl
MHPDEBUG	pick
MHPOPDEBUG	POP transactions
MHVDEBUG	window management transactions
MHWDEBUG	alternate–mailboxes

Forwarding Mail

The *forw* and *mhl* commands have two switches, '–dashmunging' and '–nodashmunging' which enable or disable the prepending of '– ' in forwarded messages. To use '–nodashmunging', you must use an *mhl* filter file.

Send

The *send* command has two switches, '–unique' and '–nounique', which are useful to certain individuals who, for obscure reasons, do not use draft–folders.

''Distribution Carbon Copy'' addresses may be specified in the *Dcc:* header. This header is removed before posting the message,and a copy of the message is distributed to each listed address. This could be considered a form of Blind Carbon Copy which is best used for sending to an address which would never reply (such as an auto–archiver).

Posting Mail

If you’re running a version of *MH* which talks directly to an *SMTP* server (or perhaps an advanced *MMDF* submit process), there are lots of interesting switches for your amusement which *send* and *post* understand:

-mail	Use the *MAIL* command (default)
-saml	Use the *SAML* command
-send	Use the *SEND* command
-soml	Use the *SOML* command
-snoop	Watch the *SMTP* transaction
-client host	Claim to be ‘‘host’’ when posting mail
-server host	Post mail with ‘‘host’’

The last switch is to be useful when *MH* resides on small workstations (or PC:s) in a network—they can post their outgoing mail with a local relay, and reduce the load on the local system. On POP client hosts, the ‘–server host’ switch is defaulted appropriately using the SMTP search–list mechanism. The *whom* command understands the last three switches.

8. CONFIGURATION OPTIONS

This manual was generated with the following configuration options in effect:

Generation Date	November 30, 1993
Primary Directory	/usr/local/
Secondary Directory	/usr/local/lib/mh/
Maildrop Location	/usr/spool/mail/$USER
Transport System	SendMail

CONTENTS

THE RAND MH MESSAGE HANDLING SYSTEM: ADMINISTRATOR'S GUIDE

UCI Version

Marshall T. Rose

First Edition:
MH Classic
(Not to be confused with a well–known soft drink)

November 30, 1993
6.8.3 #1[UCI]