In this chapter:

- *Where Do You Get UW IMAP?*
- *What Do You Get with UW IMAP?*
- *How Do You Install It?*
- *Where Can You Go for Help If You Get Stuck?*

11

Installing UW IMAP

Installing UW IMAP is easier than installing your average MS-Windows package. If you've got a few essential skills, you should not only be able to easily pull down the current UW package, but upgrade it as frequently as you desire.

With UW, compatibility is the name of the game. Unadulterated, UW IMAP runs on a wide variety of hardware and talks to an equally wide variety of mailstores. Chances are, you can just compile the thing, slap the necessary entries into your */etc/services* and */etc/inetd.conf* files, restart *inetd,* and you're in business. On a good day you may be able to go from no IMAP to a completely installed UW IMAP server inside 15 or 20 minutes.

Where Do You Get UW IMAP?

The procedures we outline in this section are for retrieving, compiling, and installing the latest source code to UW IMAP. Before we launch into that, it bears mentioning that you *can* get prebuilt binaries of the UW server, we just don't recommend it. It's our experience that just about everything runs cleaner if you build it either on the system on which it will eventually run or on one very nearly like it.

If you insist, however, prebuilt binaries are available at *ftp://ftp.cac.washington.edu/pine/unix-bin/* as *imapd-bin.<machine_type>* (where <machine_type> is probably one of *aix, du, hpux, linux, next, sgi, solaris, sun,* or *ultrix*).

Further discussion of what kind of prebuilt binaries are available may be found at *ftp://ftp.cac.washington.edu/pine/README.*

Eager Linux types can, of course, find an appropriate RPM* file at *http://rufus.w3.org/linux/RPM/Server_Mail.html*, download it over a speedy Internet connection, and install it on their machine inside of a minute or so. Once you've gotten your sea legs with UW IMAP, however, you'll probably find it worth your while to rebuild from source.

When you do build from source, the obvious first thing you'll need will be the UW IMAP source distribution. The canonical URL for the most recent version is *ftp://ftp.cac.washington.edu/imap/imap.tar.Z*. If you find that link stale, your two next best bets are to check the UW IMAP Information Center at *http://www.washington.edu/imap/* or The IMAP Connection at *http://www.imap.org/* for pointers to the right link.

Download the latest version of the IMAP source distribution using anonymous FTP. Put the distribution *tar* file in the working directory in which you will build the software, then decompress and unpack the *tar* file:

```
% zcat imap.tar.Z|tar xvf -
```

Note that if you've got the Lynx browser installed, you can save a few steps by downloading the IMAP distribution using the *lynx –dump* command:†

```
% lynx -dump ftp://ftp.cac.washington.edu/imap/imap.tar.Z | uncompress | tar xvf -
```

In the directory where you unpacked the software, you should have a top-level directory named for the current version of the IMAP package:

```
% ls -ld imap-*
drwxr-xr-x    9 104      wheel        4096 May 23 08:35 imap-4.7c
```

Inside that directory, you'll find a README file; a documentation directory; makefiles for Windows NT, Windows NT with Kerberos, Windows CE, and Unix; the source tree; and a tools directory for tools generated as part of the build process:

```
% cd imap-4.5
% ls -FaCs
total 64            4 CONTENTS       2 docs/          4 makefile.wce
   2 ./            28 Makefile       6 makefile.nt    2 src/
   2 ../            6 README         6 makefile.ntk   2 tools/
```

* RPM stands for Red Hat Package Manager. Like the Carnegie Mellon University *depot* system, it's a widely used method for retrieving software and other files from remote archives and installing them locally. To date, although RPM is attempting wider acceptance and is an open and flexible enough architecture to do so, its primary following is concentrated in the Linux community.

† At-length proselytization about the way-cool all-around utility of Lynx is somewhat beyond the scope of this chapter. Lynx is available from *http://lynx.browser.org/*. *lynx –dump* is rather like */usr/bin/cat* for the Web. It's a pretty groovy way of scriptifying or just simplifying downloads from remote archives like web or FTP sites.

Keeping Current

You'll want to keep current on which release and version of UW IMAP is the latest and what features it has. Before going any further, we should explain the numbering system for IMAP release and version numbers. Release numbers take the form MAJOR [.MINOR] [.STATUS], where MAJOR is the major version number, MINOR is the minor version number, and STATUS is an optional status for prereleases. STATUS is usually either blank (meaning final), beta, or alpha. Version numbers likewise take the form MAJOR [.EDIT]. MAJOR is the major version number, and EDIT is a number that is incremented each time the *imapd.c* source file is edited. Changes in the release number indicate that a change could have taken place anywhere in the UW IMAP Toolkit. Changes in the version number indicate that a change in *imapd.c* only has taken place.

If you run across a bug, you'll probably want to check the latest release to see if the bug has been addressed. Barring bug fixes, whether or not you want to upgrade to a newer version of UW IMAP probably depends on the feature set of the revision in question. A good way to compare revisions is to check the contents of *docs/RELNOTES* in the distribution. If a change hasn't made it into that document, check the archives of the C-Client mailing list. More on that list later.

There are two ways to find out the version of a given distribution of the UW IMAP server. One is to install it, then *telnet* into the port onto which you've installed the server and see what the prompt says.

Here's an example of finding Version 12.250 on the local host:

```
% telnet localhost imap
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
* OK localhost IMAP4rev1 v12.250 server ready
. logout
* BYE nec.unt.edu IMAP4rev1 server terminating connection
. OK Completed
Connection closed by foreign host.
```

and Version 10.231 on a UW host:

```
% telnet ftp.cac.washington.edu imap
Trying 140.142.4.227...
Connected to ftp.cac.washington.edu.
Escape character is '^]'.
* OK ftp2.cac.washington.edu IMAP4rev1 v10.231 server ready
. logout
* BYE ftp2.cac.washington.edu IMAP4rev1 server terminating connection
. OK Completed
Connection closed by foreign host.
```

A less strenuous method would be to search through *src/imapd/imapd.c* in your distribution and look for the line containing "*version". This line:

```
char *version = "12.250";       /* version number of this server */
```

comes from a 12.250 version.

Significant new versions are always announced to the IMAP Interest List. We'll give you various ways to participate in this list later on. First, let's get down to building some software.

What Systems Does It Support?

UW IMAP supports many platforms. With little or no modification, the UW software will build on:

A/UX
AIX (generic; 3.2; 4.1; AIX/370; 2.2.1 for RT)
Altos SVR4 (generic; with GCC)
AmigaDOS (with a 68020+; using AS225R2; with a 680x0 using "new" socket library)
AOS for RT
Atari ST Mint
BSD (generic; BSD/i386 3.0 and higher; FreeBSD; OpenBSD; NetBSD)
Bull DPX/2 B.O.S.
Convex
Data General DG/UX (up to 5.4)
Dynix, EP/IX
HP-UX (9.x with and without GCC; 10.x with and without GCC; 10.x with DCE security)
ICL DRS/NX
Interactive Systems OS
Linux (with Pluggable Authentication Modules (PAM) and with traditional passwords and `crypt()` in the C library)
LynxOS
MachTen, NEXTSTEP (3.x;)
OSF/1(Digital Unix) (Version 4)
PTX, Pyramid, SCO Open Server (5.0.x GCC 2.7.1*)
Solaris using GCC
Sun OS using GCC
RISC Ultrix (DEC-5000) using GCC

* Crispin notes in the Makefile: "(95q4 from Skunkware _not_ 98q2!)". He tells us that he doesn't know what that means, but probably it's meaningful to an SCO Open Server site.

What Hardware?

You name it! Everything from mainframes running AIX/370; PCs running all flavors of BSD, Linux, SolarisX86, NextStep, or SCO Open Server; Macintoshes running A/UX, and, of course, the large numbers of native Unix machines running Solaris, AIX, HP/UX, Sun OS, etc. Efforts are even under way to complete a port to Windows CE, just in case you want to run an IMAP server in your pocket![*]

What Else Do You Need?

You'll need a compiler.[†] That's about it. Of course, your IMAP server would be rather atypical if it didn't have a mailstore to serve, but strictly speaking, there's nothing to keep you from using it without a mailstore. In fact, one possible configuration might be to use a single UW IMAP server as a kind of multiplexer to other IMAP and NNTP servers. Although the UW server doesn't directly support this functionality as distributed, as the output from the CAPABILITIES command might lead you to believe, accommodations are made for those who want to add their own code to provide this function. We'll discuss those a little later on in the configuration section.

What Do You Get with UW IMAP?

After building the UW IMAP server, you will find the following programs in the source tree:

mtest
: The C-Client test bed program. This is a fairly simple command-line testing program for IMAP servers. We'll get into the specifics of using this utility in the UW system administration chapter. *mtest* is largely useful to developers using the C-Client library who want to test new mailbox drivers, system administrators troubleshooting an IMAP client/server issue who want an IMAP-specific protocol analyzer, or anyone who simply wants to learn more about what happens "on the wire" between an IMAP client and server.

ipop2d and ipop3d
: These are Post Office Protocol servers (Versions 2 and 3, respectively) that you can run on your mail server to provide POP access to any IMAP server. If a

---

[*] WinCE will potentially be run on desktop machines as well, but saying "I have an IMAP socket in my pocket 'cause I thought it was cool" just has a certain ring to it.

[†] If you're considering any other compiler, especially a C compiler, give GCC a shot first. It compiles programs in C, C++, Objective C, Ada 95, Fortran 77, and Pascal, and runs on a variety of systems ranging the vast majority of Unix and Unix-like systems to Windows NT/95 and VMS. See *http://gcc.gnu.org/* for more information.

given INBOX is accessed by the POP daemon before IMAP, then IMAP will access it in read-only mode. If IMAP gets to it first, POP access will fail with an access error.

imapd
: The IMAP4rev1 daemon. The likely reason why you're mucking about with all this stuff in the first place. A perfectly acceptable course of action for many folks with nothing-out-of-the-ordinary Unix boxes is to build the software, throw a reference to it in */etc/inetd.conf*, and be up and running.

The C-Client library (c-client.a)
: The heart of the IMAP package. All the other software in this package uses the C-Client library, and it's the API this library provides that gives UW IMAP its mailbox flexibility and extensibility.

Important Documents

The README in the top-level directory boils down the Unix install to five steps. If you've installed *inetd* services and built open source software before, you can use that quick-start list or the slightly less terse version in *docs/BUILD* to bring everything up in five minutes or so.

If you stray from the minimalist path, you're likely to find that some of the features aren't as well documented as you'd like. A post to the C-Client mailing list is likely to get a speedy answer, usually from Crispin himself. Many of the folks on C-Client like it to remain a fairly low-traffic list, however, so try to exhaust the local docs and the mailing list archives before you post. Here's an annotated list of the local docs you'll have in your archive:

CONTENTS
: An overview of what you'll find where in the IMAP Toolkit tree.

README
: Quick build notes and miscellaneous notes.

docs/BUILD
: Notes on building the Toolkit for most major platforms.

docs/drivers.txt
: Definitely one of the more valuable docs in this Toolkit. It walks you through the heuristics C-Client goes through in deciding what mailbox driver to use on any given mailbox; it also documents the UW IMAP namespace and compares the performance of each of the mailbox drivers.

*docs/rfc/**
: Copies of pertinent, and some impertinent, RFCs.

docs/CONFIG
: Directions on how to stray from the plug-and-play path by changing the code and adding authenticators and mailbox drivers.

docs/calendar.txt
: This is the best smart-ass answer to a naive question that we've run across. Worth stopping your installation, getting a cup of coffee or chai tea, and reading at leisure.

docs/RELNOTES
: Probable very old release notes dating back to the last incremental release, not the current version. Good for historical information, but cull through the C-Client archives for canonical information about your particular version of the UW IMAP Toolkit.

docs/bugs.txt
: Same caveat as for *RELNOTES*. Useful, but dated, information. The C-Client archives are a good source for information to augment this.

docs/naming.txt
: An overview of what you'll find where in the IMAP Toolkit tree.

docs/imaprc.txt
: Documentation on the use and dangers of the UW-specific *imaprc* file. A better name would be "Pandora's box." We advise you not to use this file, as does Mark Crispin, if you're not actually installing UW IMAP at the University of Washington. This file is reserved for the UW-specific black box mode, which makes numerous, undocumented assumptions about the character and arrangement of your mailstore. Use of this file is the easiest way to take what is essentially a plug-and-play effortless server install and turn it into a downward-spiraling conundrum of sysadmin pain.

docs/ internal.txt
: Concise and complete documentation on C-Client API internals. A great starting point if you're itching to write a mailbox driver.*

How Do You Install It?

The quick-start steps for the impatient are as follows. Later on, we'll cover deviations you might like to make in this routine:

1. Change directories to the top-level directory created when you untarred the distribution file. With IMAP 4.5, the directory name is *imap-4.5*.

* Actually, basing a mailbox driver on an existing, working driver is a better place to start, but you'll want to have this file handy nevertheless.

2. Select your OS/compiler combination from the list in *Makefile*. For example, *gso* is Solaris and GCC.
3. Do a *make <keyword>* where <keyword> is your OS/compiler combo. For example, *make gso*.
4. Rejoice in the wonder of compiling open source software as your IMAP Toolkit builds (alternatively, try to figure out why the compile failed or why your compiler doesn't work, etc.) Once it finishes, you'll have all the components built and ready to put into action.
5. Become *root* and copy the *imapd* binary to */usr/local/etc*. Linux admins may prefer */usr/local/sbin*.
6. Make sure you've got an entry like this in your */etc/services* file:

```
imap            143/tcp
```

7. Make sure you've got corresponding entries in your */etc/inetd.conf* file. If you're *not* using *TCP Wrappers*:*

```
imap   stream  tcp   nowait  root   /usr/local/etc/imapd   imapd
```

   If you are using *TCP Wrappers*:†

```
imap  stream  tcp  nowait  root  /usr/local/etc/tcpd  /usr/local/sbin.imapd
```

   It doesn't matter if you call your services *imap*, *imap4*, or even *InternetMail-AccessProtocol* as long as the name you use in */etc/services* is exactly the same as the name you use in */etc/inetd.conf*. Before you send a HUP signal to *inetd*, take a minute and make absolutely sure that the location you give for your *imapd* software is accurate and contains either symbolic links to the right binaries or the binaries themselves.
8. Finally, signal *inetd* to reread its configuration file:

```
# ps -ef | grep inetd
root   170     1  0   Sep 27 ?        0:00 /usr/sbin/inetd -s
# kill -HUP 170
```

The IMAP distribution includes POP2 and POP3 daemons. If you don't intend to provide POP services, there's no need to spend time installing the IPOP2d/IPOP3D servers. If you see mention of POP in the IMAP documentation, just ignore it. POP is not required for IMAP to function. Once upon a time, it made terrific sense to provide POP3 mailbox service instead of or at least in parallel to IMAP, because few mainstream mail user agents supported IMAP. That hasn't been true for a long

---

* TCP Wrappers (*ftp://ftp.win.tue.nl/pub/security*) is a wrapper program for *inetd* that allows monitoring and filtering of *inetd* services.

† If you have inexplicable problems connecting to your IMAP server from any other machine and you're running TCP Wrappers, try removing TCP Wrappers from the equation to see if that makes any difference.

time. All major email apps now support IMAP, and POP3 is becoming a legacy protocol. If you've got some stubborn POP3 users, though, or some special need, UW IMAP's POP2 and POP3 daemons are a pretty elegant way to gateway your new mailstore using an old protocol. POP2 is such a legacy protocol, however, that unless you have clear user demand, there's little or no reason to install the POP2 daemon.

Not Leaving Well Enough Alone…

There's a small handful of things you can do to alter the baseline installation of the IMAP toolkit and accompanying servers. In most cases, you won't need to alter anything in the default installation. You might be well advised to just run it in the default configuration for a while and see what customer demands are met by modest changes in the installation, rather than trying to anticipate the demands of your users in advance.

Monkeying with the Makefiles

One way of customizing your installation is to change certain variables in the makefiles. Once you've made a backup copy of the original *Makefile* or a last-known working copy and you've changed the variables you want, go ahead and redo your "make <SystemType>" as discussed previously to build your customized version of the IMAP Toolkit.

The top-level *Makefile* has three lines of note:

```
EXTRAAUTHENTICATORS=
EXTRADRIVERS=mbox
PASSWDTYPE=std
```

As you might guess, `EXTRAAUTHENTICATORS` and `EXTRADRIVERS` are the mechanisms through which additional authentication and mailbox drivers are added into the C-Client library and, by extension, the IMAP daemon. We'll go into more detail about this in Chapter 12, *UW System Administration*. Notice also that the *mbox* driver is explicitly enabled. All other drivers are enabled by default, there's no need to list them here. These lines are primarily for implementing add-on authenticators and mailbox drivers.

The `PASSWDTYPE` variable can be set to one of a variety of plaintext password mechanisms, which are listed out in the makefile:

```
# afs   AFS authentication database
# dce   DCE authentication database
# nul   no plaintext authentication (note: this will break some secure
#        authenticators -- don't use without checking first!!)
# std   system standard (typically passwd file), determined by port
```

There are a handful of extra switches you can turn on in the top-level makefile, like Y4K (yep, Year 4000) leap-year correction to accommodate the fact that years evenly divisible by 4000 aren't leap years, or enabling British Summer Time. They're all well-documented; all you need to do is uncomment the appropriate line for each switch.

Another Makefile you may want to examine and change is the *imapd/Makefile*. There are likewise three variables you may want to customize:

```
ALERT=/etc/imapd.alert
USERALERT=.imapalert
ANO=/etc/anonymous.newsgroups
```

`ALERT` specifies the general IMAP alert file. Place a single-line message in this file, and each user should see that notification as soon as they start using the IMAP server. Different clients should use different ways of displaying the message, but all compliant clients should display the message somehow. `USERALERT` is a file you put in the user's home directory (or whatever you define their *homedir* to be in the code) to send a notification to an individual user. `ALERT` is good for such notifications as "IMAP will be going down for three hours starting 3am November 4, 2001," while `USERALERT` is good for such things as "Attention user jbk00234: Your E-Mail is obnoxious, please get a life at once!"

The `ANO` variable defines a file that, if it exists, enables anonymous mailbox access on the IMAP server. More about this in Chapter 12.

Clobbering the Code

The vast majority of code-level changes that you're likely to want to make to the way UW IMAP operates have been isolated to the file *src/osdep/unix/env_unix.c*. `env_init()` and `sysinbox()` are routines that each have a value that you may want to change to customize to your environment. The source documentation also says that `mailboxdir()` and `mailboxfile()` may also have some commonly useful things to change, but that hasn't been our experience. If you want to investigate those, the details are in *docs/CONFIG*.

`env_init()`

If you want the top-level mailbox directory for UW to be something other than the user's home directory, search in *env_unix.c* for this line:

```
myHomeDir = cpystr (home);/* use real home directory */
```

and change these lines to something like the following. In this example, you would change the default top-level mailbox directory to *~/mail* for any given user:

```
sprintf (tmp,"%s/mail",home);
      myHomeDir = cpystr (tmp);
```

One source of confusion for new UW IMAP users is suddenly seeing all their home directory files mixed in with their mailboxes. Setting `myHomeDir` to a different value fixes this.

`sysinbox()`

On some systems, it might be preferable to have the MTA deliver incoming mail to somewhere different from the traditional */var/spool/mail/USER* location. If you have a good reason to do this, say because you want incoming mail to be subject to the same partition quota as the stored mailboxes, make a change in the `sysinbox()` routine.

This example supposes that, instead of the traditional mailspool, you want INBOX to be physically located in *~/.mailbox* in a user's home directory. In this case, you would change:

```
sprintf (tmp,"%s/%s",MAILSPOOL,myusername ());
```

to:

```
sprintf (tmp,"%s/.mailbox",myhomedir ());
```

and your user INBOXes would be distributed across the home directory space.

Where Can You Go for Help If You Get Stuck?

Two good starting places are The IMAP Connection, at *http://www.imap.org/,* and the University of Washington IMAP Information Center, at *http://www.washington.edu/imap/*. Each site is a portal to various sites and archives. The IMAP Connection focuses primarily on the IMAP protocol and the software that uses it. Possibly its best resource is an interactive database of IMAP software: client, server, gateway, utilities, etc. The UW IMAP Information Center is primarily focused on UW IMAP software.

Speaking of archives, two mailing lists essential for getting good, up-to-date information are the IMAP mailing list and the C-Client mailing list. Instructions for joining and viewing the archives of the IMAP mailing list are at *http://www.washington.edu/imap/imap-list.html*. Information about the C-Client list and its archives is at *http://www.washington.edu/imap/c-client-list.html*.

Finally, you can use your local Usenet server or Deja News (*http://www.deja.com/*) to read the *comp.mail.imap* group for a good unbiased, or at least equally unbalanced, collective viewpoint on the state of IMAP clients and servers.