Pine Technical Notes

Version 3.91, October 1994

PC-Pine and UNIX Pine are still under active development. Pine 3.90, released August 1994, was a major release with many new features. Pine 3.91 is a maintenance release that corrects a number of bugs reported after the 3.90 release. However, there are also two new features:

preserve-start-stop-characters
compose-rejects-unqualified-addrs

The first one was needed to cause Pine to revert to its pre-3.90 behavior for sites that use software flow-control (usually via XON and XOFF characters). Version 3.90 was changed to ignore those characters so that a mis-typed "control-S" or XOFF character would not cause Pine to mysteriously and silently freeze, but this change caused problems for sites using software flow-control. The second new feature was needed locally to reduce the probability of mis-typed addressbook nicknames resulting in mid-directed email.

Other than the above, and a few clarifications, these Tech Notes are essentially the same as the version 3.90 Tech Notes.

Section 1 – Introduction

History and Design Goals

Pine was originally conceived in 1989 as a simple, easy-to-use mailer for administrative staff at the University of Washington in Seattle. This constituency had previously been using a very nice mail system derived from UCLA's "Ben" mailer for the MVS operating system, but when the cost of maintaining our MVS system became prohibitive, we needed to find a Unix-based mailer that preserved the user-interface strengths of "Ben". Our goal was to provide a mailer that naive users could use without fear of making mistakes. We wanted to cater to users who were less interested in learning the mechanics of using electronic mail than in doing their jobs; users who perhaps had some computer anxiety. We felt the way to do this was to have a system that didn't do surprising things and provided immediate feedback on each operation; a mailer that had a limited set of carefully-selected functions.

At that time, we could not find any Unix mailer (commercial or freely available) that met our requirements. Consequently, we reluctantly concluded that we would need to develop our own. The Elm mailer seemed like a reasonable starting point since its source code was freely available, so we started modifying it. Today there is virtually no Elm code left, and Pine has evolved so that many "power-user" features may be (optionally) enabled. We have tried to remain true to our original simplicity and ease-of-use goals by providing *optional* features for sophisticated users. In fact, if none of Pine's options are enabled, the latest version has almost the same look-and-feel as the very first version.

One of the greatest problems with most mailers on Unix systems is the editor. One can normally choose between emacs and vi. We experimented with some versions of emacs and settled on a hacked version of micro emacs. Eventually it became heavily modified and tightly integrated with the rest of Pine. One of the main features of having a tightly coupled editor is that it can guide the user through editing the header of the message, and Pine takes great care to do this. A very simple and efficient interface to the Unix spell command was also added. The emacs- style key bindings were retained, though most of the other wild and wonderful emacs functions were not. The Pine composition editor is also available as a very simple stand alone editor named "pico".

Also working at the University of Washington is the original author of the Internet Message Access Protocol (IMAP). IMAP is a functional superset of POP, and provides a way to manipulate mailboxes on remote servers as if they were local. Specific advantages of IMAP over POP include: support for inbox access from multiple computers, access to more than one remote folder, selective access to MIME message parts, and support for disconnected operation.

Not long after the Pine project began, The IMAP author had finished writing the "c-client" library as an interface to IMAP and as a switch between drivers for IMAP mailboxes, Berkeley mail files and Tenex mail files. In time, "c-client" became a full messaging API with support for RFC-822 parsing, MIME parsing and decoding, SMTP and NNTP drivers, and so forth. Great care was taken to make the code writing the mail files robust against disks filling up, and inter-process locking in order to guarantee mail file consistency. It was clear that Pine would benefit greatly from using the c-client to access mail storage so the original low-level Elm code was replaced by calls to c-client library routines. Consequently Pine can write and access a variety of different mail file formats and new ones can be added by creating a simple driver. In addition the c-client does a very careful job of doing all the RFC 822 header parsing and achieves the highest compliance with the RFC.

Most of the work done on Pine from 6/92 to 6/93 focused on changes needed to support a truly distributed electronic messaging environment (e.g. remote folder manipulation), and getting Pine to run on DOS (which was a *lot* of work). The first version of PC-Pine (3.84) was released in July 1993, and included first steps toward integrating news and email access in Pine. Doing the DOS port was very difficult for a variety of reasons, but especially because of DOS memory management (or lack thereof). However, simply porting Pine 3.07 to DOS was not sufficient. For a desktop mailer such as PC-Pine to be useful at UW, it was necessary to fully support access to existing *remote* saved-message folders, as well as local (desktop) folders -- and of course, the remote INBOX. Accomplishing this required extensions to IMAP, a new version of the IMAPd server code, and extensive work in Pine to support multiple collections of folders.

The principal reason for porting Unix Pine to DOS was to obviate the need for PC users to transfer files between their PC and the Unix system running Pine. Now it is possible to save messages directly to the PC's filesystem, and to directly include PC files in outgoing messages. And with Pine's MIME capability, binary files (e.g. word processing documents, spreadsheets, image files, executables) may be directly attached to your messages.

With Pine 3.90, significant new functionality has been added, notably aggregate operations for manipulating groups of messages at once, the first (alpha) release of PC-Pine for the Winsock network interface standard, and greatly improved Usenet (News) support. One of the early interpretations of the name "Pine" was "Pine Is No-longer Elm"; today a "Program for Internet News and Email" seems more apropos.

Throughout Pine development, we have had to strike a balance between the need to include features which advanced users require and the need to keep things simple for beginning users. To strike this balance, we have tried to adhere to these design principles:

- The underlying model presented to the user has to be simple and clear. Underlying system operation is hidden as much as possible.

- It's better to have a few easily understood commands that can be repeated than to have some more sophisticated command that will do the job all at once.

- Whenever the user has to select a command, file name, address, etc., the user should be given (or can get) a menu from which to make the selection. Menus need to be complete, small, organized and well thought out.

- Pine must provide immediate feedback for the user with each operation.

- Pine must be very tolerant of user errors. Any time a user is about to perform an irreversible act (send a message, expunge messages from a folder), Pine should ask for confirmation.

- Users should be able to learn by exploration without fear of doing anything wrong. This is an important feature so the user can get started quickly without reading any manuals and so fewer manuals are required.

- The core set of Pine functions should be kept to a minimum so new users don't feel "lost" in seemingly extraneous commands and concepts.

Just as there were goals relating to the look and feel of Pine, there were equally important goals having to do with Pine's structure–the things that users never see but still rely on every time they use Pine. While Pine can be used as a stand-alone mail user agent, one of its strongest assets is its use of the Internet Message Access Protocol (IMAP) for accessing remote email folders. In addition, Pine was one of the first programs to support the Multipurpose Internet Mail Extensions (MIME) specification.

With MIME, Pine users can reliably send any binary file to any other person on the Internet who uses a MIME compliant email program.

The decision to use IMAP and MIME reflect the importance of interoperability, standardization and robustness in Pine. As you work with Pine more, you will see other features which reflect the same values. For example, Pine enforces strict compliance with RFC 822, implements a strong mail folder locking mechanism and verifies a process before overwriting any files (e.g. addressbook, expunging messages).

Pine Components

If you have picked up the Pine distribution, then you already know that Pine comes in a few different pieces. They are:

Pine This main code from which the Pine program is compiled.

Pico Pico is the name for the Pine composer. The Pico code is used in two ways: (1) it is compiled on its own to be a stand-alone editor or (2) compiled as a library for Pine to support composition of messages within Pine. Pico is Pine's internal editor invoked when users need to fill in header lines or type the text of an email message.

Imap An API for IMAP. Includes the C-Client library, which is compiled into Pine, and the IMAP server IMAPd. C-Client implements the IMAP protocol and also negotiates all access between Pine and the mail folders it operates on. The C-Client routines are used for email folder parsing and interpreting MIME messages. IMAPd is a separate server that handles IMAP connections from any IMAP-compliant email program. When Pine accesses a remote mailbox, the Pine program is the IMAP client and the IMAPd program is the IMAP server.

Section 2 – Background Details

Domain Names

Domain names are used to uniquely name each host on the Internet. A domain name has a number of parts separated by periods. Each label represents a level in the hierarchy. An example of a name is:

olive.cac.washington.edu

In this domain name the top-level label is *edu,* indicating it is at an educational institution, the second-level label is *washington,* indicating the University of Washington. *cac* is a specific department within the University of Washington, and *olive* is the host name. The top-level names are assigned by Internet organizations, and other names are assigned at the appropriate level. The Domain Name Service, DNS, is the distributed database used to look up these names.

Pine relies on domain names in multiple places. A domain name is embedded into the message-id line generated for each piece of email. A domain name is needed to contact an IMAP server to get access to remote INBOXes and folders. Most importantly, domain names are needed to construct the From: line of your outgoing messages so that people on the Internet will be able to get email back to you.

On UNIX systems, you can set the domain via the *user-domain* variable in the Pine configuration file, or rely on the file /*etc*/*hosts* which usually sets the name of the local host. While Pine can often deliver email without the domain name being properly configured, it is best to have this set right. Problems can usually be solved by adjusting the system's entry in the /*etc*/*hosts* file. The fully-qualified name should be listed before any abbreviations.

128.95.112.99   olive.cac.washington.edu   olive

is preferred over

128.95.112.99   olive   olive.cac.washington.edu

On PCs, the task of configuring the domain name is a bit different. Often times, PCs do not have domain names–they have *IP addresses*. IP addresses are the numbers which uniquely identify a computer on the network. The way you configure your IP address depends on the networking software which you use on the PC. You can refer to the documentation which came with your networking software or see the PC specific installation notes for help configuring the IP address with your network software.

With PCs, it is vital that users set the variable *user-domain* in the Pine configuration file (*PINERC*).

Details on configuring Pine with correct domain names can be found in the Domain Settings section of this document.

RFC 822 Compliance

Pine tries to adhere to RFC 822 a little more strongly than some other mailers and uses the "full name <address>" format rather than the older "address (full name)" format. The intent of the standard is that parentheses should only be for comments. Pine displays and generates the newer format, but will parse the old format and attempt to turn it into the new one.

As far as outgoing email is concerned, Pine fully-qualifies addresses whenever possible. They are even displayed in fully-qualified form on the terminal as the user composes a message. This makes addresses more clear and gives a hint to the user that the network extends beyond the local organization. Pine implements fully-qualified domain names by tacking on the local domain to all unqualified addresses which a user types in. Any address which does not contain a "@" is considered unqualified.

The newer format for addresses allow for spaces and special characters in the full name of an address. For this reason, commas are required to separate addresses. If any special characters as defined in RFC 822 appear in the full name, quotes are required around the address. Pine will insert the quotes automatically. The common cases where this happens are with periods after initials and parentheses.

Because Pine fully complies with RFC 822, it is sometimes difficult to use non-Internet address formats such as UUCP's *host!user* or DECNet's *USER::HOST* with Pine. People who run Pine on these systems have made local modifications to Pine or to the mail transport agent (e.g. sendmail) to make things work for them. Another special case that Pine does not allow for are the sites in the United Kingdom which require two "local" domains (one in the form *machine.site.ac.uk* for use outside the UK and the other *uk.ac.site.machine* for use inside the UK). This special case requires local modifications to Pine.

Pine expects dates to be in the standard RFC 822 format which is something like:

[www,] dd mmm yy hh:mm[:ss] [timezone]

It will attempt to parse dates that are not in this format. When an unparsable date is encountered it is displayed as *xxx xx* when shown in the FOLDER INDEX screen.

SMTP and Sendmail

Pine is a *user agent* not a *message transfer agent.* In plain English, that means Pine does not know how to interact with other computers on the Internet to deliver or receive email. What Pine does know how to do is help users read, organize and create email. The "dirty work" of delivering and accepting email is handled by other programs.

All outgoing email is delivered to a mail transfer program or to an SMTP server. The most common mail transfer program is *sendmail.* When Pine on a UNIX computer uses the local *sendmail,* it first writes the message to a temporary file in */tmp.* Then Pine runs a shell in the background that runs *sendmail* on the temporary file and then removes it. This is done with a shell in the background so the user doesn't have to wait for *sendmail* to finish. By default, *sendmail* is invoked with the *-t* flag to cause it to read and parse the header to determine the recipients; the *-oem* flag to cause errors to be mailed back; and the *-oi* flag to ignore dots in incoming messages. Systems administrators can choose to configure Pine to use a different mail transfer program or even *sendmail* with different flags. See the section on UNIX Pine Compile-time Options for more details on this.

Pine can also operate as an SMTP client. SMTP stands for *Simple Mail Transfer Protocol;* it specifies the rules by which computers on the Internet pass email to one another. In this case, Pine passes outgoing email messages to a designated SMTP server instead of to a mail transfer program on the local machine. A program on the server then takes care of delivering the message. To make Pine operate as an SMTP client, the *smtp-server* variable must be set to the IP address or host name of the SMTP server within your organization. This variable accepts a comma separated list of servers, so you can specify multiple SMTP servers. PC-Pine only runs as an SMTP client.

Internet Message Access Protocol (IMAP)

IMAP is a remote access protocol for message stores. Pine uses IMAP to get at messages and folders which reside on remote machines. With IMAP, all messages are kept on the server. An IMAP client (such as Pine) can request specific messages, headers, message structures, etc. The client can also issue commands which delete messages from folders on the server. IMAP's closest kin is POP, the Post Office Protocol, which works by transferring an entire mailbox to the client where all the mail is kept. For a complete comparison of IMAP and POP, see the paper .i "Comparing Two Approaches to Remote Mailbox Access: IMAP vs. POP" by Terry Gray. The paper can be found as the file doc/imap.vs.pop in the standard Pine distribution.

IMAP Features:

> Allows access to mail folders from more than one client computer.
>
> Works well over low-bandwidth lines because information is sent in small pieces as needed by the user. For example, only header information is sent to build index lists, and if someone sends a 2MB audio file via MIME, you can choose when (or if) you want to get that part of the message.
>
> Email can be delivered and stored on a well-maintained and reliable server which is "always-up".
>
> Folders can be accessed and manipulated from anywhere on the Internet.
>
> Users can get to messages stored on different folders within the same Pine session.
>
> Allows use of IMAP server for searching and parsing.
>
> The latest revision of IMAP (IMAP4) also provides for disconnected operation, including resynchronization of message state between mail servers and message caches on clients. Pine does not yet support this capability, however.

IMAP2 is defined in RFC 1176. IMAP4, the proposed revision to IMAP2, is described in the document "latest-imap-draft" in the /mail directory of ftp.cac.washington.edu. IMAP4 will be formally documented in an upcoming RFC. Pine 3.90 is an IMAP2bis client, but does not yet implement all of the IMAP4 extensions. (IMAP2bis was an interim specification superseded by IMAP4.) Pine takes advantage of the extensions defined in IMAP2bis for efficient and selective access to MIME body parts. We expect the next major release of Pine (probably version 4.0) to be fully compatible with the IMAP4 specification.

Multipurpose Internet Mail Extensions (MIME)

MIME is a way of encoding a multipart message structure into a standard Internet email message. The parts may be nested and may be of seven different types: Text, Audio, Image, Video, Message, Application and Multipart (nested). The MIME specification allows email programs such as Pine to reliably and simply exchange binary data (images, spreadsheets, etc.) MIME includes support for international character sets, tagging each part of a message with the character set it is written in, and providing 7-bit encoding of 8-bit character sets. It also provides a simple rich text format for marking text as bold, underlined, and so on. There is a mechanism for splitting messages into multiple parts and reassembling them at the receiving end.

The MIME standard was officially published in June of 1992 as RFC 1341 and subsequently revised in RFC 1521 when it became a full Internet Standard. Pine 3.0 was one of the first email programs to Implement MIME. Now, there are dozens of commercial and freely available MIME-capable email programs. In addition, MIME is being added to newsreaders so MIME messages can be posted and read in USENET newsgroups.

An actual MIME message looks something like this:

```
From lgl@olive.cac.washington.edu Tue Jul 14 17:55:17 1992
Date: Tue, 14 Jul 1992 17:55:17 -0700 (PDT)
From: Laurence Lundblade <lgl@cac.washington.edu>
Subject: Test MIME message
To: Laurence Lundblade <lgl@cac.washington.edu>

--16820115-1435684063-711161753:#2306
Content-Type: TEXT/PLAIN; charset=US-ASCII

The text of the message would go here. It is readable if
one doesn't mind wading around a little bit of the MIME
formatting. After this is a binary file in base 64
encoding. It has been shortened for this example. The
Base 64 stuff looks dorky in PostScript because
troff -me doesn't have a fixed font like courier.

Laurence Lundblade
  lgl@cac.washington.edu
    Computing and Communications, University of Washington

--16820115-1435684063-711161753:#2306
Content-Type: TEXT/plain; name=login
Content-Transfer-Encoding: BASE64
Content-Description: NeXT login program
```


```
AYAAAABAAAAAQAAAAQAAAL4AAAAAQAAAAEAAAJYAAAAAAAAAAAA
AAAAAAAABfsAAADFAAAFswAAAAHAAAABwAAAAgAAAAAX190ZXh0
AAAAF9fVEVYVAAAAAAAAAAAAAAAAAAAAAAAQpAAAAxQAAAABAAAA
AAAAAAAAAAAAABfX2Z2bWxpYl9pbml0MAAAX19URVhUAAAAAAAA
KQAAAEwAAATuAAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAF9fZnZt
XQxAABfX1RFWFQAAAAAAAAAAAAAAAAAR1AAAAAAAAABToAAAAAgAA
AAAAAAAAAAAAAAAX19jc3RyaW5nAAAAAAAAAAF9fVEVYVAAAAAAA
BHUAAADQQAAFOgAAAAAAAAAAAAAAAAAAAAACAAAAAAAAAAABfX2Nv
AAAAAAAX19URVhUAAAAAAAAAAAAAAAAAFRgAAACsAAAYLAAAAAIA
AAAAAAAAAAAAAAAAF9fZGF0YQAAAAAAAAAAAAAABfX0RBVEEAAAAA
AAVxAAAAQgAABjYAAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAX19i
AAAAAAAAF9fREFUQQAAAAAAAAAAAAAAAAABbMAAAADAAAAAAAAAAC
AAAAAABAAAAAAAAAAABfX2NvbW1vbgAAAAAAAAAAAX19EQVRBAAAA
CAlcwAlZCBMT0dJTiBGQUlMVVJFJXMgT04gJXMsICVzAHN1AGxv
Wxsb2Mgb3V0IG9mIG1lbW9yeQoAJXMgdG9vIGxvbmcNCgAvZXRj
3Vzci9hZG0vd3RtcAAAAABAKCMpUFJPR1JBTTpsb2dpbiAgUFJP
WRzLTQyICBERVZFTE9QRVI6cm9vdCAgQlVJTFQ6U3VuIE5vdiAx
zoyMSBQU1QgMTk5MAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAQCgjKSBDb3B5cmlnaHQgKGMp
DE5ODcsIDE5ODggVGhlIFJlZ2VudHMgb2YgdGhlIFVuaXZlcnNp
2FsaWZvcm5pYS4KIEFsbCByaWdodHMgcmVzZXJ2ZWQuCgBAKCMp
wk1LjQwIChCZXJrZWxleSkgNS85Lzg5AAAAABHUAAAR1f//////
wAAEdQAABHUAAAR1AAAEdQAAAEsAxwREwT/GhkSDxcWAAAR2gAA
```

AAR5gAAEeoAABHuAAAR8gAAEfYAABH6AAAR/gAAEgIAABIGAAAA
AAB

--16820115-1435684063-711161753:#2306--

For more information about MIME, see RFC 1521 or the FAQ in the newsgroup comp.mail.mime or the paper *MIME Overview* by Mark Grand. You can find the paper via ftp on adad.premenos.sf.ca.us as pub/mime.ps or /pub/mime.txt. For details about Pine's implementation of MIME, see the two MIME sections later in this document.

Folder Collections

Folder Collections are Pine's way of dealing with more than a single group of folders. With advent of PC-Pine and the development of tools within IMAP to better manage remote folders, the time was ripe to provide a mechanism for defining a group of remote folders. PC-Pine forced the issue in that many potential PC-Pine users would be migrating from UNIX pine in a time-sharing environment and, thus, would have some investment in their archived messages on that host.

Currently, Pine has no way to dynamically create or define collections, but there is much work still going on in this area. The hope is to provide a general way to define, display and navigate remote folder collections in a consistent way across platforms and applications. Especially important to this goal will be the hierarchy support provisions in the IMAP4 specification. Stay tuned!

For a more complete description of Folder Collections, see the section on "Syntax for Collections".

Section 3 – Building and Installation

The Pine distribution is designed to require as little configuration and effort at compile time as possible. Still, there are some Pine behaviors which are set at the time you compile Pine. For each of these, there is a reasonable (our opinion) default built into the code, so most systems administrators will have no need for these steps.

UNIX Pine Compile-time Options

The files you may need to modify are *./pine/makefile.xxx* and *./pine/osdep/os-xxx.h* where "xxx" is the 3-letter code for your platform. You can give the command *build help* to see the list of ports incorporated into Pine and their associated 3-letter codes. The file *./pine/makefile.xxx* is where you would set your compiler options. By default, Pine will be compiled with debugging on, optimization and profile off. Note that if you compile with DEBUG off, then Pine will not create its normal debug files, no matter how the debug-level and debug command line flag are set.

Most of Pine's behaviors are set in the file *./pine/osdep/os-xxx.h,* which includes comments that explain each setting. Some of these can only be set when you compile. Others, however, can be overridden by command-line flags to Pine or settings in Pine's user or system configuration files. Some of the options which can be set when compiling:

> USE_QUOTAS: Determines whether quotas are checked on startup. Default is to not check the quota.
>
> ALLOW_CHANGING_FROM: Determines whether users are allowed to modify the From line on outgoing mail. Even with this turned on, users will have to include From in their *default-composer-hdrs* or *customized-hdrs* in order to be able to edit the From line. Default is to not allow any changing.
>
> DEFAULT_DEBUG: Sets the level of debugging output created in Pine's debug files. Default is level 2.
>
> NEW_MAIL_TIME: Interval between new-mail checks. Default is 150 seconds.
>
> OVERLAP: Number of lines overlap when user views the next page of a message. Default is 2 lines.
>
> USE_TERMINFO: Instructs Pine to use the terminfo database instead of termcap. Default varies by system.
>
> SENDMAIL and SENDMAILFLAGS: Sets the name and flags for the local program that will be called to handle outgoing email. Default is */usr/lib/sendmail -oi -oem -t.*
>
> SYSTEM_PINERC: The name of the file which holds Pine configuration information for all users on the system. Default on UNIX systems is */usr/local/lib/pine.conf.*
>
> SYSTEM_PINERC_FIXED: The name of the file which holds the same type of information as for SYSTEM_PINERC, but only for variables that the administrator wants to keep fixed. That is, users are not allowed to change variables that are specified in the FIXED file. Default on UNIX systems is */usr/local/lib/pine.conf.fixed.*

There are a couple of more obscure options which are in the source code because a few people have asked for them or because we changed our minds about them being a good idea in general.

ENCODE_FROMS: Use Quoted-printable encoding so that *From's* at the beginning of lines don't end up being escaped by >'s. Most people seem to dislike the Q-P encoding more than the > escapes so this is off by default. Once everyone has MIME mail readers, we'll turn this on by default.

NO_KEYBOARD_LOCK: Disable the keyboard locking function in the main menu. Keyboard locking is enabled by default. (Keyboard lock may also be turned off by adding "disable-kblock-cmd" to the feature list variable in the global pine.conf file.)

Pico Compile-time Options

There are even fewer options needed when compiling Pico. The two interesting ones are for UNIX Pico versions only. The file that may need some changing is *./pico/os_unix.h.* Whatever is set will effect the behavior of the Pico stand-alone program as well as the composer within Pine.

SPELLER: Names the program called to do "normal" spell-checking.

TERMCAP and TERMINFO: Determines which of these terminal databases will be used.

IMAPd Compile-time Options

There are no options or settings required for the version of IMAPd distributed with Pine. If you need to be doing more complex modifications to IMAP, then you should pick up the IMAP development package and work with that code. The developer's version of IMAP is available for anonymous ftp from *ftp.cac.washington.edu* in the directory *mail*. The file is called *imap.tar.Z.*

Building the Pine Programs

You may have already compiled Pine and tried it out. If so, great! If not, you should be able to do it without too much trouble by following these step-by-step instructions:

1. Figure out what platform you're building for. You can give the command *build help* to see the list of ports incorporated into Pine. What you need is the three letter code for the platform. Some examples are *nxt* for the Next operating system and *ult* for Ultrix. If your platform is not in the list of ports, then you might have some work ahead of you. First, check the file *doc/pine-ports.* to see if there are others working on a port for your platform or to see if the port is included in the "contrib" section of the source code. Ports in the *contrib* directory were contributed by Pine administrators from around the world, but the Pine development team has not been able to test the code. If Pine has not yet been ported to your platform at all, read the section on Porting Pine in this document.

2. Make sure you're in the root of the Pine source. When you type *ls* you should see the following files and directories (or something close to it):

README	build	doc	makefile	pine
bin	contrib	imap	pico	

3. Make sure you're getting a clean start by giving the command *build clean.* This should take only a few seconds to run.

4. Give the command *build xxx* where *xxx* is the three letter code you picked in step 1. The compiler should grind away for a few minutes.

5. When the compilation is complete the sizes of the four binaries built (pine, mtest, imapd, pico) will be displayed. The actual binaries are in the various source directories. In addition, the *bin* directory contains a link to each program compiled. You can just copy them out of *bin* or try them from there.

Installing Pine and Pico on UNIX Platforms

Installing Pine and Pico is remarkably simple. You take the program files which you have just transferred or built and you move them to the correct directory on your system. Most often the binaries go in */usr/local/bin* though sometimes they are placed in */usr/bin.* All the help text is compiled into Pine so there are no **required** auxiliary files.

There are, however, three optional auxiliary files: */usr/local/lib/pine.info,* */usr/local/lib/pine.conf,* and */usr/local/lib/pine.conf.fixed.* The file *pine.info* contains text on how to get further help on the local system. It is presented as the first page of the help text for the main menu and should probably refer to the local help desk or the system administrator. If this file doesn't exist a generic version which suggests ''talking to the computer support staff at your site'' is shown. The file *pine.conf* is used to set system-wide default configurations for Pine. The file *pine.conf.fixed* is also used to set system-wide default configurations for Pine. The difference between these two files is that configuration variables set in the *pine.conf.fixed* file may not normally be over-ridden by a user. See the section on Pine Configuration later in this document for details about the *pine.conf* and *pine.conf.fixed* files.

Installing PC-Pine

Beginning with the Pine 3.90 release, there is a PC-Pine version that runs under windows using the Winsock network interface. For those who still need to run the DOS version of PC-Pine, there are versions for four different TCP/IP network stacks: FTP Inc's PC/TCP, Novell's LAN Workplace for DOS, Sun's PC/NFS, and WATTCP for packet drivers. PC-Pine needs to be able to interact closely with the stack loaded on your PC. Most of the time, this occurs automatically. However, there are certain modifications that need be made.

LAN Workplace for DOS Version 4.1

Set the environment variable *EXCELAN* in the PC's *AUTOEXEC.BAT* file. This provides the necessary links so that LAN Workplace for DOS 4.1 can translate domain names to IP numbers correctly. It is needed because Pine was developed for LAN Workplace 4.0 and this particular variable is treated differently in 4.1 than in 4.0. The *EXCELAN* variable must point to the directory in which LAN Workplace is installed.

PC/TCP versions before 2.2

You need a file called *PCTCP.INI* which contains a bare-minimum 2-line description of the PC's configuration. It looks like this:

[pctcp ifcust 0]
ip-address=*xx.xx.xx.xx*

Where *xx.xx.xx.xx* is the IP address of the PC. Pine also requires an environment variable, *PCTCP,* which points to this file. For example:

set PCTCP=C:\PINE\PCTCP.INI

Packet Drivers

Pine needs to be made aware of the PC's network configuration file. Simply edit the file *WATTCP.CFG* included in the Pine distribution. The file includes 5 configuration settings—IP-address, gateway, netmask, nameserver(s) and domainslist. If you have a network configuration file for NCSA Telnet then *WATTCP.CFG* is just a pared down version of the *CONFIG.TEL* file you already made. Take a look at *CONFIG.TEL* to find the correct settings for *WATTCP.CFG*. Once the configuration file is made, the DOS environment variable *WATTCP.CFG* needs to point at it. For example:

set WATTCP.CFG=C:\PINE

In addition to networking software issues, you might need to worry about setting the time zone. PC-Pine includes the time zone as part of outgoing email. There is a generic way for PC applications to get the time zone, but, because PC-Pine is one of a very few applications which requires this information, time zone might not be previously configured.

The trick is to add an environment variable, *TZ,* to your PC's *AUTOEXEC.BAT* file. The format for the *TZ* environment variable is as follows:

ZZZ[+H]H[:MM:SSTTT]

First is the 3-letter code for your standard time, then a "+" or a "-" for direction of offset from GMT, then the amount of offset (hours, minutes, seconds) and finally the 3-letter code for your summer- or daylight savings time. Everything in [] brackets is optional.

The default time zone is "PST-8PDT" (U.S. Pacific Time). Coincidentally, Microsoft is headquartered in that time zone.

As an example, people in the Eastern part of the US should add this line to their *AUTOEXEC.BAT* files:

TZ=EST-5EDT

Installing IMAPd

When the Pine distribution is built on a UNIX station, the IMAP server binary, *imapd,* is compiled. Installing *imapd* requires placing the binary in the appropriate directory, usually */usr/etc,* and adding entries to */etc/services* and */etc/inetd.conf* or their counterparts. The following line is appropriate for */etc/services:*

imap 143/tcp # Mail transfer

and the next line is appropriate for */etc/inetd.conf:*

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

The */etc/inetd.conf* file entry may vary on different versions of UNIX. Some have a slightly different set of fields. Also the pathname in */etc/inetd.conf* must match the path where *imapd* is installed.

With this configuration, the IMAP server runs without pre-authentication. Each new IMAP connection requires a correct username and password. IMAP can also be run with pre-authentication based on the standard *rsh* mechanism. To enable this, the user account on the IMAP server must contain a valid file which grants access to the client machine. Enabling *rimap* authentication is done by creating a link called */etc/rimapd* to *imapd.* Basically, what is happening is that Pine is taking advantage of the ability that *rsh* has to use privileged TCP ports so it doesn't have to run in privileged mode. If the *rimap* authentication fails it will drop back to plain password authentication.

PC-Pine cannot take advantage of *rimap* authentication. Also, if your system uses a distributed configuration database, like NIS, Yellow Pages or Netinfo, be sure that appropriate steps are taken to ensure the above mentioned information is updated.

Support Files and Environment Variables: UNIX Pine

This section lists the various files which Pine uses which are not email folders. All of these are the default names of files, they may vary based on Pine's configuration.

/usr/local/lib/pine.conf
: Pine's global configuration file.

/usr/local/lib/pine.conf.fixed
: Non-overridable global configuration file.

/usr/local/lib/pine.info
: Local pointer to system administrator.

~/.pinerc
: Personal configuration file for each user.

~/.addressbook
: Personal addressbook

~/.addressbook.lu
: Personal address book lookup file (index file to speed up lookups).

~/.newsrc
: Personal USENET subscription list. This is shared with other newsreading programs.

~/.pine-debugX
: The files created for debugging Pine problems. By default, there are 4 .pine-debug files kept at any time.

~/.signature
: A signature file which will be included in all outgoing email messages.

~/.pine-interrupted-mail
: The text of a message which was interrupted by some unexpected error which Pine detected.

~/mail/postponed-msgs
: A folder of messages which the user chose to postpone.

/etc/mailcap
: System-wide mail capabilities file. Only used if $MAILCAPS not set.

~/.mailcap
: Personal mail capabilities file. Combines with system-wide mailcap. Only used if $MAILCAPS not set.

The location of the following support files may be controlled by variables in the personal or global Pine configuration file: signature, addressbook and its index file, postponed messages, and newsrc.

Unix Pine uses the following environment variables:

TERM
: Tells Pine what kind of terminal is being used.

DISPLAY
: Determines if Pine will try to display IMAGE attachments.

SHELL
: If not set, default is /bin/sh

MAILCAPS
: A semicolon delimited list of path names to mailcap files.

Support Files and Environment Variables: PC-Pine

This section lists the various files which PC-Pine uses which are not normal mail folders. All of these are the default names of files, they may vary based on Pine's configuration.

<PINE.EXE directory>\PINE.HLP
: File containing Pine's internal help text.

<PINE.EXE directory>\PINE.NDX
: Index of Pine's help text used by PC-Pine to locate entries.

$PINERC or $HOME\PINE\PINERC or <PINE.EXE dir>\PINERC
: Path to (required) personal configuration file.

$PINECONF
: Path of optional global configuration file.

<PINERC directory>\ADDRBOOK
: Personal addressbook

<PINERC directory>\ADDRBOOK.LU
: Personal address book lookup file (index file to speed up lookups).

<PINERC directory>\PINE.SIG
: A signature file which will be included in all outgoing email messages.

<PINERC directory>\PINE.PWD
: A file containing encrypted password for remote mail server.

<PINERC directory>\PINEDEBG.TXT
: Location of Pine debug file.

<PINERC directory>\MAILCAP and/or <PINE.EXE dir>\MAILCAP
: These paths are only used if $MAILCAPS not set.

$HOME\NEWSRC or <PINERC directory>\NEWSRC
: Personal USENET subscription list. This may be shared with other newsreading programs.

$HOME\MAIL\INTRUPTD
: The text of a message which was interrupted by some unexpected error which Pine detected.

$HOME\MAIL\POSTPOND
: A folder of messages which the user chose to postpone.

PC-Pine's help text and help text index file are expected to reside in the same directory as the *PINE.EXE* executable, as they are essentially extensions of the executable. The personal configuration file may be in the same directory as the executable, or if that is inconvenient because the executable is on a shared or read-only drive, then it can be in a file named by the *$PINERC* environment variable, or in *$HOME\PINE\PINERC*, where if not set, *$HOME* defaults to the root of the current working drive.

Most of the other support files key off of the location of the *PINERC* file. However, in the case of the NEWSRC file, the path *$HOME\NEWSRC* is checked first. Also, the postponed messages and interrupted message folders are placed in the default folder collection, normally in the directory *$HOME\MAIL*.

The location of the following support files may be controlled by variables in the personal or global Pine configuration file: signature, addressbook (and its index file), postponed messages, and newsrc.

PC-Pine uses the following environment variables:

PINERC
: Overrides default path to pinerc file.

PINECONF
: Optional path to global pine config file.

HOME
: If not set, Pine uses the root of the current drive, e.g. C:.ip TMP or TEMP Specifies location of temporary storage area

COMSPEC
: Specifies shell for external commands.

MAILCAPS
: A semicolon delimited list of path names to mailcap files.

Section 4 – Command Line Arguments

Pine and PC-Pine can accept quite a few command-line arguments. Many of these arguments overlap with variables in the Pine configuration file. If there is a difference, then a flag set in the command line takes precedence. Both Pine and PC-Pine expect command line arguments to be preceded by the "-" (dash) as normally used by UNIX programs.

[address]
: Send-to: If you put an unqualified string (or strings) in the command line, Pine reads them as email addresses. Pine will startup in the composer with a message started to the person/people specified. Once the message is sent, the Pine session closes.

-a
: Special anonymous mode for UWIN*.

-conf
: Configuration: Prints a sample system configuration file to the screen or standard output.

-create_lu *addrbook sort-order*
: Create auxiliary index (LookUp) file for *addrbook* and sort *addrbook* in *sort-order*, which may be *dont-sort*, *nickname*, *fullname*, *nickname-with-lists-last*, or *fullname-with-lists-last*. Only useful when creating global or shared address books.

-d *debug-level*
: Debug Level: Sets the level of debugging information written by Pine. *debug-level* can be set to any integer 0-9. A debug level of 0 turns off debugging for the session. (Actually there are some levels higher than 9, now, but you probably don't want to see them.)

-f *folder*
: Startup folder: Pine will open this folder in place of the standard INBOX.

-F *file*
: Open named text file and view with Pine's browser.

-h
: Help: Prints the list of available command-line arguments to the screen.

-i
: Pine will start up in the FOLDER INDEX screen instead of the MAIN MENU. Configuration equivalent: *initial-keystroke-list=i*

-I *a,b,c,...*
: Initial Keystrokes: Pine will execute this comma-separated sequence of commands upon startup. This allows users to get Pine to start in any of its menus/screens. You cannot include any input to the composer in the initial keystrokes. The key <Return> is represented by a ''CR'' in the keystroke list; the spacebar is designated by the letters ''SPACE''. Control keys are two character sequences beginning with ''^'', such as ''^I''. A tab character is ''TAB''. Function keys are ''F1'' - ''F12'' and the arrow keys are ''UP'', ''DOWN'', ''LEFT'', and ''RIGHT''. Configuration equivalent: *initial-keystroke-list*

-k Function-Key Mode: When invoked in this way, Pine expects the input of commands to be function-keys. Otherwise, commands are linked to the regular character keys. Configuration equivalent: *use-function-keys* included in *feature-list.*

-l Folder-List: With "-l" set, Pine will default to an expanded folder list. This means that the FOLDER LIST screen will always show all folders in all collections. Default is to show the folders in the current collection only. Configuration equivalent: *expanded-view-of-folders* in *feature-list.*

-n *n* Message-Number: When specified, Pine starts up in the FOLDER INDEX screen with the current message being the designated message number.

-nr Special mode for UWIN*.

-o *folder*
Opens the INBOX (or a folder specified via the -f argument) ReadOnly.

-p *file* Uses the named file as the personal configuration file instead of ~/*pinerc* or the default PINERC search sequence PC-Pine uses.

-P *file* Uses the named file as the system wide configuration file instead of /*usr*/*local*/*lib*/*pine.conf*. UNIX Pine only.

-pinerc *file*
Output fresh pinerc configuration to *file*, preserving the settings of variables that the user has made. Use *file* set to ''-'' to make output go to standard out.

-r Restricted Mode: For UNIX Pine only. Pine in restricted mode can only send email to itself. Save and export are limited.

-sort *key*
Sort-Key: Specifies the order messages will be displayed in for the FOLDER INDEX screen. *Key* can have the following values: subject, arrival, date, from, size, subject/reverse, arrival/reverse, date/reverse, from/reverse, size/reverse. The default value is "arrival". The *key* value reverse is equivalent to arrival/reverse. This option will be expanded in the future to allow sorting on "to" and "cc". Configuration equivalent: *sort-key*.

-z Enable Suspend: When run with this flag, the key sequence ctrl-z will suspend the Pine session. Configuration equivalent: *enable-suspend* included in *feature-list*.

-*option=value*
Assign *value* to the config option *option.* For example, *-signature-file=sig1* or -*feature-list=signature-at-bottom* (Note: feature-list values are additive).

* UWIN = University of Washington Information Navigator

Section 5 – Configuration and Preferences

Pine Configuration

There is very little in Pine which **requires** compile-time configuration. In most cases, the compiled-in preferences will suit users and administrators just fine. When running Pine on a UNIX system, the default built-in configuration can be changed by setting variables in the system configuration file, */usr/local/lib/pine.conf*. Both Pine and PC-Pine also use personal (user-based) configuration files. On UNIX machines, the personal configuration file is the file *~/.pinerc*. For PC-Pine systems, the personal configuration file is in *$PINERC* or *$HOME\PINE\PINERC* or *<PINE.EXE dir>\PINERC*.

The syntax of a non-list configuration variable is this:

<variable> = <value>

If the value is absent then the variable is unset. To set a variable to the empty value the syntax is "". This is equivalent to an absent value except that it overrides any system-wide value that may be set. Quotes may be used around any value. All values are strings and end at the end of the line or the closing quote. Leading and trailing space is ignored unless it is included in the quotes. For some variables the only appropriate values are *yes* and *no.* There is also a second type of variable, lists. A list is a comma-separated list of values. The syntax for a list is:

<variable> = <value> [, <value> , ...]

A list can be continued on subsequent lines by beginning the line with white-space. Both the per-user and global configuration files may contain comments which are lines beginning with a #.

For UNIX Pine, there are five ways in which a variable can be set. In decreasing order of precedence they are:

(1) the system-wide *fixed* configuration file
(2) a command line argument
(3) the personal configuration file (which is usually set from the Config screen)
(4) the system-wide configuration file
(5) default in the source code.

So, system-wide fixed settings always take precedence over command line flags, which take precedence over per-user settings, which take precedence over system-wide configuration settings, which take precedence over source code defaults. PC-Pine has the same precedence, but it does not use a system-wide fixed configuration file.

You may get a sample/fresh copy of the system configuration file by running *pine -conf.* The result will be printed on the standard output with short comments describing each variable. (The online help in the Setup/Config screen provides longer comments.) If you need to fix some of the configuration variables, you would use the same template for the fixed configuration file as for the regular system-wide configuration file. (If it isn't clear, the purpose of the fixed configuration file is to allow system administrators to restrict the configurability of Pine. It is by no means a bullet-proof method.) Pine will automatically create the personal configuration file the first time it is run, so there is no need to generate a sample. Pine reads and writes the personal configuration file occasionally during normal operation. Users will not normally look at their personal configuration file, but will use the Setup/Config screen from within Pine to set the values in this file. If a user does add additional comments to the personal configuration file they will be retained. Pine always writes this file at least once when running, so you can tell when a user last invoked Pine by checking the date on this file.

References to environment variables may be included in the Pine configuration files. The format is *$variable* or *${variable}*. The character ˜ will be expanded to the *$HOME* environment variable.

When environment variables are used for Pine settings which take lists, you must have an environment variable set for each member of the list. That is, Pine won't properly recognize an environment variable which is set equal to a comma-delimited list. It is OK to reference unset environment variables in the Pine configuration file, which will expand to nothing.

General Configuration Variables

The following is a list of all Pine configuration variables, in alphabetical order. Note that not all variables apply to all versions of Pine and that some variables are only applicable in a system configuration file and some are only applicable in a personal configuration file.

addrbook-sort-rule

This variable sets up the default address book sorting. Currently, Pine will accept the values *dont-sort*, *fullname-with-lists-last*, *fullname*, *nickname-with-lists-last*, and *nickname*. The default is to sort by fullname with lists last.

address-book

A list of personal address books. Each entry in the list is an optional nickname followed by a pathname or file name relative to the home directory. This list will be added to the *global-address-book* list to arrive at the complete set of address books.

bugs-nickname, bugs-fullname and bugs-address

System-wide configuration file only. These are used by the Report Bug command.

character-set

This sets the character set used by the terminal. Currently appropriate values are US-ASCII, ISO-8859-1 through ISO-8859-9 and ISO-2022-JP. See the section on international character sets for more details. The default is US-ASCII.

customized-hdrs

Add these custom headers when composing. Also possible to add default values to these custom headers or to any of the standard headers. This is a list variable. Each entry in the list is a header name (the actual header name that will appear in the message) followed by an optional colon and value. For example, if a Reply-to header was needed because it was different from the From address, that could be accomplished with:

customized-hdrs=Reply-to: fred_flintstone@bedrock.net

default-composer-hdrs

Show only these headers (by default) when composing a message. This list may include headers defined in the *customized-hdrs* list.

default-fcc

The name of the folder to which all outgoing mail goes is set here. The compiled-in default is *sent-mail* (UNIX) or *sentmail* (PC). It can be set to "" (two double quotes with nothing between them) to turn off saving copies of outgoing mail. If the default-fcc is a relative file name, then it is relative to your default collection for saves (see *folder-collections*).

editor

UNIX Pine only. Sets the name of the alternate editor for composing mail (message text only, not headers). It will be invoked with the "^_" command or it will be invoked automatically if the *enable-alternate-editor-implicitly* feature is set.

fcc-name-rule

Determines default folder name for fcc when composing. Currently, Pine will accept the values *default-fcc*, *by-recipient*, or *last-fcc-used*. If set to *default-fcc*, then Pine will use the value defined in the *default-fcc* variable (which itself has a default) for the Fcc header field. If set to *by-recipient*, then Pine will use the name of the recipient as a folder name for the fcc. The relevant recipient is the first address in the To field. If set to "last-fcc-used", then Pine will offer to fcc to whatever folder you used previously. In all cases, the field can still be edited after it is initially assigned. If the fcc field in the address book is set for the first To address, that value over-rides any value derived from this rule.

feature-list

This is a list of features (options) which may be turned on. You may also turn features off (the default) by prepending the characters *no-* to any of the features. The *feature-list* is additive. That is, first the system-wide *feature-list* is read and then the user's *feature-list* is read. This makes it possible for the system manager to turn some of the features on by default while still allowing the user to cancel that default. However, some of the documentation assumes that all of the features are off by default, so use this with care. In Unix Pine, features can be individually fixed on or off by setting the feature on or off in the system-wide *fixed* configuration file. Descriptions are omitted here. See the online help for descriptions of each feature (in the Setup/Config screen). Exception: the four *disable-* features are intentionally suppressed from the Config screen, as they are intended for use by system administrators in the system-wide fixed config file. Their meaning should be self-explanatory. Also the *use-function-keys* feature is hidden from the config screen. Here is the current list of possible features.

assume-slow-link
auto-move-read-msgs
auto-open-next-unread
compose-rejects-unqualified-addrs
compose-sets-newsgroup-without-confirm
delete-skips-deleted
disable-keyboard-lock-cmd
disable-config-cmd
disable-password-cmd
disable-update-cmd
enable-aggregate-command-set
enable-alternate-editor-cmd
enable-alternate-editor-implicitly
enable-bounce-cmd

enable-flag-cmd
enable-full-header-cmd
enable-incoming-folders
enable-jump-shortcut
enable-mail-check-cue
enable-suspend
enable-tab-completion
enable-unix-pipe-cmd
expanded-view-of-addressbooks
expanded-view-of-folders
expunge-without-confirm
include-attachments-in-reply
include-header-in-reply
include-text-in-reply
news-post-without-validation
news-read-in-newsrc-order
preserve-start-stop-characters
quit-without-confirm
save-will-quote-leading-froms
save-will-not-delete
save-will-advance
select-without-confirm
show-selected-in-boldface
signature-at-bottom
use-current-dir
use-function-keys
user-lookup-even-if-domain-mismatch

folder-collections

This is a list of one or more collections where saved mail is stored. See the sections describing folder collections and collection syntax for more information. The first collection in this list is the default collection for saves, including *default-fcc*'s.

font-name

Winsock version of PC Pine only.

font-size

Winsock version of PC Pine only.

global-address-book

A list of shared address books. Each entry in the list is an optional nickname followed by a pathname or file name relative to the home directory. This list will be added to the *address-book* list to arrive at the complete set of address books. Global address books are defined to be readonly.

image-viewer

This variable names the program to call for displaying parts of a MIME message that are of type IMAGE. If your system supports the *mailcap* system, you don't need to set this variable.

inbox-path

This specifies the name of the folder to use for the INBOX. Normally this is unset so the system's default is used. The most common reason for setting this is to open an IMAP mailbox for the INBOX. For example, {*imap5.u.example.edu*}*inbox* will open the user's standard *INBOX* on the mail server, imap5.

incoming-folders

This is a list of one or more folders other than *INBOX* that may receive new messages. This list is slightly special in that it is always expanded in the folder lister. In the future, it may become more special. For example, it would be nice if Pine would monitor the folders in this list for new mail.

initial-keystroke-list

This is a comma-separated list of keystrokes which Pine executes on startup. Items in the list are usually just characters, but there are some special values. *SPACE, TAB,* and *CR* mean a space character, tab character, and a carriage return, respectively. *F1* through *F12* stand for the twelve function keys. *UP, DOWN, LEFT,* and *RIGHT* stand for the arrow keys. Control characters are represented with ^<*char*>. A restriction is that you can't mix function keys and character keys in this list even though you can, in some cases, mix them when running Pine. A user can always use only *character* keys in the startup list even if he or she is using *function* keys normally, or vice versa.

last-time-prune-questioned

Personal configuration file only. This variable records the month the user was last asked if his or her sent-mail folders should be pruned. The format is *yy.mm*. This is automatically updated by Pine when the the pruning is done or declined. If a user wanted to make Pine stop asking this question he or she could set this time to something far in the future.

last-version-used

Personal configuration file only. This is set automatically by Pine. It is used to keep track of the last version of Pine that was run by the user. Whenever the version number increases, a new version message is printed out.

mail-directory

This variable was more important in previous versions of Pine. Now it is used only as the default for storing personal folders (and only if there are no *folder-collections* defined). The default value is ˜/*mail* on UNIX and *$HOME\MAIL* on a PC.

news-collections

This is a list of collections where news folders are located. See the section describing collections for more information.

nntp-server

One or more NNTP servers (host name or IP address) which Pine will use for outgoing news. If you read and post news to and from a single NNTP server, you can get away with only setting the *nntp-server* variable and leaving the *news-collections* variable unset.

normal-background-color

PC-Pine only. Currently, Pine will accept the colors *black*, *blue*, *green*, *cyan*, *red*, *magenta*, *yellow*, or *white*.

normal-foreground-color
PC-Pine only. See *normal-background-color* for possible colors.

personal-name
Personal configuration file only (not applicable in global config. file). User's full personal name. On UNIX systems, the default is taken from the accounts data base (/etc/passwd).

personal-print-command
UNIX personal configuration file only. This corresponds to item 3 in the printer menu. This variable retains the value of *personal-print-command* when the printer is set to something other than item 3. The *personal-print-command* can be set within Pine using the printer setup menu.

postponed-folder
The folder where postponed messages are stored. The default is postponed-msgs (Unix) or POSTPOND (PC).

printer
UNIX Pine only. This is the current setting for a user's printer. This variable is set from Pine's printer-setup function. The value must be either

"attached-to-ansi" -or-
the value of *personal-print-command* -or-
the value of *standard-printer* from the system-wide configuration

read-message-folder
If set, mail in the *INBOX* that has been read but not deleted is moved here, or rather, the user is asked whether or not he or she wants to move it here upon quitting Pine.

reverse-background-color
PC-Pine only. See *normal-background-color* for possible colors.

reverse-foreground-color
PC-Pine only. See *normal-background-color* for possible colors.

saved-msg-name-rule
Determines default folder name when saving. Currently, Pine will accept the values "default-folder", "by-sender", "by-from", "by-recipient", or "last-folder-used". If set to *default-folder*, then Pine will offer the folder "saved-messages" (UNIX) or "SAVEMAIL" (PC) for saving messages. If set to *by-from*, then Pine will offer to save the message in a folder with the same name as the From, if there is one, or the Sender otherwise. If set to *by-sender*, then Pine will offer to save the message in a folder with the same name as the Sender, if there is one, or the From otherwise. If set to *by-recipient*, then Pine will offer to save the message in a folder with the same name as the recipient, which is the newsgroup if this was sent to a newsgroup or the To address if not. If set to "last-folder-used", then Pine will offer to save in whatever folder you used previously.

signature-file

Names the file to be included as the signature. This defaults to ˜/*.signature* on UNIX and *<PINERC directory>\PINE.SIG* on a PC.

smtp-server

One or more SMTP servers (host name or IP address) which Pine will use for outgoing mail. If not set, Pine passes outgoing email to the *sendmail* program on the local machine. PC-Pine users must have this variable set in order to send mail as they have no *sendmail* program.

sort-key

This variable sets up the default index sorting. The default is to sort by arrival order. It has the same functionality as the *-sort* command line argument and the *$* command in the folder index. If a *sort-key* is set, then all folders open during the session will have that as the default sort order.

standard-printer

System-wide configuration file only. Specifies the command for printer selection number 2 on the printer menu. Unix only.

use-only-domain-name

Can be set to *yes* or *no.* At this point anything but *yes* means *no.* If set to *yes* the first label in the host name will be lopped off to get the domain name and the domain name will be used for outgoing mail and such. That is, if the host name is *carson.u.example.edu* and this variable is set to *yes,* then *u.example.edu* will be used on outgoing mail. Only meaningful if *user-domain* is NOT set.

user-domain

Sets the domain or host name for the user, overriding the system host or domain name. See the domain name section.

user-id

PC-Pine only. Sets the username that is placed on all outgoing messages.

window-position

Winsock version of PC Pine only. Window position in the format: CxR+X+Yn Where C and R are the window size in characters and X and Y are the screen position of the top left corner of the window.

Retired Variables

Variables that are no longer used by the current Pine version. When an obsolete variable is encountered, its value is applied to any new corresponding setting and a comment is place before it noting that it is no longer in used. The replaced values at the time of this document include:

elm-style-save

Replaced by *saved-msg-name-rule*

feature-level

Replaced by *feature-list.*

header-in-reply
Replaced by *include-header-in-reply* in the *feature-list.*

old-style-reply
Replaced by *signature-at-bottom* in the *feature-list.*

save-by-sender
Replaced by *saved-msg-name-rule.*

show-all-characters
No replacement, it always works this way now.

Pine in Function Key Mode

The standard Pine uses alphabetic keys for most commands, and control keys in the composer. Despite possible appearances, the current bindings are the result of much discussion and thought. All the commands in the composer are single control characters. This keeps things very neat and simple for users. Two character commands in the composer are a possibility, but we're trying to avoid them because of the added complexity for the user.

Pine can also operate in a function-key mode. To go into this mode invoke *pine -k* or (on some UNIX systems) *pinef.* On a UNIX system, you can link or copy the *pine* executable to *pinef* to install *pinef.* Alternatively, users and systems administrators can set the *use-function-keys* feature in the personal or system-wide Pine configuration file. The command menus at the bottom of the screen will show *F1-F12* instead of the alphabetic commands. In addition, the help screens will be written in terms of function keys and not alphabetic keys.

One of the results of using Pine in function-key mode is that users can only choose from twelve commands at any given time. In alphabetic-key mode, a user can press a key for a command (say, q to quit) and that command can be fulfilled. In function-key mode, the command must be visible on the bottom key-menu in order to be used. There are some screens where 34 commands are operational; function-key users can get to all of them, just not all at once.

Domain Settings

Pine uses the default domain for a few different tasks. First, it is tacked onto the user-id for outgoing email. Second, it is tacked onto all "local" (unqualified) addresses in the "To:" or "Cc:" fields of messages being composed (unless they are found in the address book). The domain name is also used to generate message-id lines for each outgoing message and to allow Pine to check if an address is that of the current Pine user.

Pine determines the domain name according to whichever of these it finds. The list here is in decreasing order of precedence.

(1) Value of the variable *user-domain* in the system fixed configuration file

(2) Value of the variable *user-domain* in the personal configuration file

(3) Value of the variable *user-domain* is the system-wide configuration file

(4) Value from an external database (DNS, */etc/hosts*, NIS) as modified by a system fixed configuration file if *use-domain-name-only* set to "yes"

(5) Value from an external database (DNS, /*etc*/*hosts*, NIS) as modified by a personal configuration file if *use-domain-name-only* set to "yes"

(6) Value from an external database (DNS, /*etc*/*hosts*, NIS) as modified by a system configuration file if *use-domain-name-only* set to "yes"

(7) Unmodified value (host name) from an external database

The easiest way for this system to work is for PC-Pine users and UNIX Pine system administrators to set the *user-domain* variable. The variable *use-only-domain-name* is helpful if your site supports/requires hostless addressing, but for some reason you don't want to use the *user-domain* variable.

A new feature in 3.90 is called *user-lookup-even-if-domain-mismatch*. This will cause the personal name field to be looked up from the password file even if the domain of an address isn't a substring of the local host name. See the online help in the Setup/Config screen for full information.

Syntax for Collections

In many environments, it is quite common to have collections of archived mail on various hosts around the network. Using the folder collections facility in Pine, access to these archives is just as simple as access to folders on Pine's local disk.

"Collection" is the word we use in Pine to describe a set of folders. A collection corresponds loosely to a "directory" containing mail folders. Folders within a defined collection can be manipulated (opened, saved-to, etc) using just their simple name. Any number of folder collections can be defined, and pine will adjust its menus and prompts to help navigate them.

The way collections are defined in Pine is with the *folder-collections* variable in the Pine configuration file. *Folder-collections* takes a list of one or more collections, each (optionally) preceded by a user-defined logical name (label). Once collections are defined, Pine adjusts its menus and behavior to allow choosing files by their simple name within the collection. Collections are always defined in the configuration file; there is no time that Pine will ever ask a question which requires a user to input a collection specifier. This might change in the future if/when the Goto command is extended to allow jumping to a collection/directory as well as an individual folder.

Consider the following:

```
folder-collections=   Local-Mail     C:\MAIL\[],
                      Remote-Mail    {imap.u.example.edu}mail/[]
```

The example shows two collections defined (a comma separated list; newlines in the list are OK if there's one or more spaces before the next entry), one local and one remote. Each collection is a space-delimited pair of elements–first an optional logical-name and second the collection specifier. The logical-name can have spaces if it has quotes around it (but keeping the logical name short and descriptive works best). Pine will use the logical-name (if provided) to reference all folders in the collection, so the user never has to see the ugliness of the collection specifier.

The collection specifier can be thought of as an extended IMAP format (see the "Remote Folders" section for a description of IMAP format names). Basically, a pair of square-brackets are placed in the fully qualified IMAP path where the simple folder name (the part without the host name and path) would appear. Like IMAP, the path can be either fully qualified (i.e., with a leading '/') or relative to your home directory.

An advanced feature of this notation is that a pattern within the square brackets allows the user to define a collection to be a subset of a directory. For example, a collection defined with the specifier:

```
M-Mail          C:MAIL/[m*]
```

will provide a view in the folder lister of all folders in the PC's "C:MAIL" directory that start with the letter 'm' (case insensitive under DOS, of course). Further, the wildcard matching will honor characters trailing the '*' in the pattern.

From within Pine, the FOLDER LIST display will be adjusted to allow browsing of the folders in any defined collection. Even more, you'll notice in the Goto and Save commands a pair of sub-commands to rotate through the list of logical collection names, so only a simple name need be input in order to operate on a folder in any collection.

The first collection specified in the *folder-collections* has special significance. That folder is the "default collection for saves". In cases where the user does not specify which collection should be used to save a message, the default collection for saves will be used. Also, if the *default-fcc* is a relative file name, then it is relative to the default collection for saves.

The notion of collections encompasses both email folders and news reading. The variable *news-collections* uses nearly the same format as *folder-collections*. Newsgroups can be defined for convenient access via either IMAP or NNTP. There are advantages and disadvantages to both access methods. In the IMAP case, your news environment state is maintained on the server and, thus, will be seen by any client. The downside is that, at the moment, you must have an account on the server. In the NNTP case, server access is mostly anonymous and no state/accounting need be maintained on it. The downside is that each client, for now, must individually maintain news environment state.

An example pinerc entry might be:

```
news-collections=   Remote-State    *{news.u.example.edu}[],
                    Local-State     *{news.u.example.edu/nntp}[]
```

Note that each news collection must be preceded by a '*' to indicate non-mail access. Only newsgroups to which you are subscribed are included in the collection.

The pattern matching facility can be applied so as to define a news collection which is a subset of all the newsgroups you subscribe to. For example, this could be a valid collection:

```
                    Newsfeed-News  *{news.u.example.edu/nntp}[clari.*]
```

Collection handling is a tough problem to solve in a general way, and the explanation of the syntax is a bit ugly. The upside is, hopefully, that for a little complexity in the Pine configuration file you get simple management of multiple folders in diverse locations.

Syntax for Remote Folders

Remote folders are distinguished from local folders by a leading host name bracketed by '{' and '}'. The path and folder name immediately following the closing bracket,

'}', is interpreted by the IMAP server and is in a form compatible with that server (i.e., path delimiters and naming syntax relative to that server).

Typically, a folder name without any path description is understood to reside in the user's "home directory" (i.e., in some way the user's personal, writable file area), as are incomplete path designations. However, the IMAP specification does not require that unqualified folder names live in one's home directory, so some IMAP servers may require fully qualified names. An example of a remote folder specification would be,

{mailhost.cac.washington.edu}mail/saved-messages

This example simply specifies a folder named ''saved-messages'' on the imap server ''mailhost.cac.washington.edu'', in the ''mail'' subdirectory of the user's home directory. Easy isn't it?

To confuse things a bit, qualifiers are permitted within the brackets following the host name. These qualifiers consist of a slash, '/' character followed by a keyword or keyword and value equality, and have the effect of modifying how the connection is made to the host specified. An example of such a specification might be,

*{pine.cac.washington.edu/anonymous}updates

Another example might be,

*{news.u.washington.edu/nntp}comp.mail.mime

Both of these examples illustrate a different qualifier. The first, specifying ''anonymous'' access to the IMAP server on ''pine.cac.washington.edu''. The second is interesting in that it specifies an altogether different access method: access via the Network News Transport Protocol (NNTP). Both examples bring to light one remaining subtlety. The leading ''*'' tells pine to treat the remote folder as a Bulletin-Board (i.e., typically a shared, read-only resource) and to adjusts its behavior accordingly.

Sorting a Folder

The mail index may be sorted by subject, size, sender, date, or arrival order. Each sort order can also be reversed. The *$* command will prompt the user for the sort order. The sort order can also be specified on the command line with the *-sort* flag or (equivalently) with the *sort-key* variable in the *.pinerc* file. When a user changes folders, the sort order will go back to the original sort order. The command line (*-sort*) or configuration file sort specification (*sort-key*) changes the original sort order.

When a folder is sorted and new mail arrives in the folder it will be inserted in its properly sorted place. This can be a little odd when the folder is sorted by something like the subject. It can also be a little slow if you are viewing a large, sorted INBOX, since the INBOX will have to be re-sorted whenever new mail arrives.

The sorts are all independent of case and ignore leading or trailing white space. There are actually two forms of subject sort. One called "Subject" and the other called "OrderedSubj". They both ignore "Re:" at the beginning and "(fwd)" at the end of the subjects. Subject sorts all the subjects alphabetically. OrderedSubj sorts by subjects alphabetically, groups messages with the same subject (pseudo-threads), then sorts the groups by the date of the first message of the group. The sort by sender sorts by the userid, not the full name. The arrival sort is basically no sort at all and the date sort depends on the format of the date. Some dates are in strange formats and are unparsable. The time zone is also taken into account.

Sorting large mail folders can be very slow since it requires fetching all the headers of the mail messages. With UNIX Pine, only the first sort is slow since Pine keeps a copy of all the headers. One exception is sorting in reverse arrival order. This is fast because no headers have to be examined. Pine will show progress as it is sorting.

Alternate Editor

In the Pine composer you can use any text editor, such as *vi* or *emacs*, for composing the message text. The addresses and subject still must be edited using the standard Pine composer. If you include the feature *enable-alternate-editor-cmd* in your *.pinerc* you can type ^_ while in the body of the message in the composer and be prompted for the editor. If you also set the *editor* variable in your *.pinerc* then ^_ will invoke the configured editor when you type it.

Turning on the feature *enable-alternate-editor-implicitly* will automatically invoke the editor you have defined with the *editor* variable whenever you enter the body of a message you are composing. For example, when you move out of the last header line and into the body of the message, the alternate editor will be automatically invoked.

We know that many people would like to use the alternate editor to edit the mail header as well. We considered several designs for this and didn't come up with one that we liked and that was easy to implement. One of the main problems is that you lose access to the address book.

Signatures and Signature Placement

If the file ~/*.signature* (UNIX) or <*PINERC*directory>\PINE.SIG (PC) exists, it will be included in all outgoing messages. It is included before composition starts so that the user has a chance to edit it out if he or she likes. The file name for the signature can be changed by setting the *signature-file* variable in the *.pinerc.* There is no way to have Pine include different signatures in different outgoing messages automatically. You can do this by hand, however, by having multiple signature files (*.sig1*, *.sig2*, *.sig3*, *etc*) and choosing to include (^R in the composer) the correct one for the message being sent.

Pine's default behavior encourages a user to put his or her contribution before the inclusion of the original text of the message being forwarded or replied to, This is contrary to some conventions, but makes the conversation more readable when a long original message is included in a reply for context. The reader doesn't have to scroll through the original text that he or she has probably already seen to find the new text. If the reader wishes to see the old message(s), the reader can scroll further into the message. Users who prefer to add their input at the end of a message should set the *signature-at-bottom* feature in the *feature-list*. The signature will then be appended to the end of the message after any included text. This feature applies when replying, not when forwarding.

Feature List Variable

Pine used to have *feature levels* for users with different amounts of experience. We found that this was too restrictive. Pine now has a *feature-list* instead. Each user may pick and choose which features they would like enabled (simple to do in the Setup/Config screen). There is a short on-line help explaining the effect of each of the features in the Setup/Config screen. When the cursor is highlighting a feature, the "?" command will show the help text for that feature. Features don't have values, they are just turned on or off. They are all off by default.

The *feature-list* variable is different from all other configuration variables in that its value is additive. That is, the system-wide configuration file can have some features turned on by default. The user can select other features in their personal configuration file and those features will be **added** to the set of features turned on in the system-wide configuration file. (With all other configuration variables, the user's values **replace** the system-wide values.) Likewise, additional features may be set on the command-line with the argument "-feature-list=". These will be added to the others.

The treatment of *feature-list* in the system-wide *fixed* configuration file is also different from other variables. The system management can fix the value of individual features by placing them in the fixed configuration file. Users will not be able to alter those features, but will still be able to set the other non-restricted features the way they like.

Because *feature-list* is additive, there is a way to turn features off as well as on. Prepending the prefix "no-" to any feature sets it to off. This is useful for over-riding the system-wide default in the personal configuration file or for over-riding the system-wide default or the personal configuration value on the command line. For example, if the system-wide default configuration has the *quit-without-confirm* feature set, the user can over-ride that (and turn it off) by including *no-quit-without-confirm* in the personal configuration file or by giving the command line argument *-feature-list=no-quit-without-confirm.* More features (options) will no doubt continue to be added.

Additional Notes on PC-Pine

Below are a few odds and ends worth mentioning about PC-Pine. They have to do with DOS-specific behavior that is either necessary or useful (and sometimes both!).

As PC-Pine runs in an environment with limited access control, accounting or auditing, an additional line is automatically inserted into the header of mail messages generated by PC-Pine:

X-Sender: <userid>@<imap.host>

By popular demand of system administrators, PC-Pine has been modified to prevent sending messages until the user has successfully logged into a remote mail server. Even though PC-Pine cannot prevent users from changing the apparent identity of the sender of a message, the IMAP server login name host name included in the X-Sender line provide some level of traceability by the recipient. However, this should not be considered a rigorous form of authentication. It is extremely lightweight, and is not a replacement for true authentication.

Hand in hand with authentication and accounting is user information. Since PC-Pine has no user database to consult for *user-id*, *personal-name*, etc., necessary information must be provided by the user/installer before PC-Pine can properly construct the "From" address required for outbound messages. PC-Pine will, by default, prompt for the requisite pieces as they are needed. This information corresponds to the *PINERC* variables *user-id*, *personal-name*, *user-domain*, and *smtp-server*.

The user is then asked whether or not this information should automatically be saved to the *PINERC*. This is useful behavior in general, but can lead to problems in a lab or other shared environment. Hence, these prompts and automatic saving of configuration can be turned off on an entry by entry basis by setting any of the above values in the *PINERC* to the null string (i.e., a pair of double quotes). This means that the user will be prompted for the information once during each pine session, and no opportunity to save them in the *PINERC* will be offered.

Along similar lines, a feature allowing automatic login to the imap-server containing the user's *INBOX* has also been requested. This feature is not enabled by default, but requires the existence of the file named *PINE.PWD* in the same directory as the *PINERC*. Even with the existence of this file, the user must still acknowledge a prompt before the password is saved to the file. If PC-Pine is configured to access several different IMAP servers, each password entered will be kept (associated with the corresponding host name) in memory during the current session, and optionally, in the *PINE.PWD* file for use in subsequent sessions.

WARNING! Use this feature with caution! It effectively makes the user's mail no more secure than the physical security of the machine running PC-Pine. What's more, while the password is cloaked by a mild (some might say, feeble) encryption scheme, it is nonetheless sitting in a file on the PC's disk and subject to cracking by anyone with access to it. *BEWARE*!

Another feature of DOS is the lack of standard scratch area for temporary files. During the course of a session, PC-Pine may require numerous temporary files (large message texts, various caches, etc.). Where to create them can be a problem, particularly when running under certain network operating systems. PC-Pine observes the *TMP* and *TEMP* environment variables, and creates temporary files in the directory specified by either. In their absence, PC-Pine creates these files in the root of the current working drive.

Section 6–Behind the Scenes

Many people ask how certain Pine features are implemented. This section outlines some of the details.

Address Books

The address book file is named, by default, *.addressbook* in the user's Unix home directory, or in the case of PC-Pine, *ADDRBOOK*, in the save directory as the *PINERC* file. There may be more than one address book, and the default name can be over-ridden via an entry in any of the Pine configuration files. The two configuration variables *address-book* and *global-address-book* are used to specify the file names of the address books. Each of these variables is a list variable. The total set of address books for a user is the combination of all the address books specified in these two lists. Each entry in the list is an optional nickname followed by a file name. The nickname is everything up to the last space before the file name. The *global-address-book* list will typically be configured in the system-wide configuration file, though a user may over-ride it like most other variables. Address books which are listed in the *global-address-book* variable are forced read-only, and are typically shared among multiple users.

Address books are simple text files with lines in the format:

<nickname>TAB<fullname>TAB<address>TAB<fcc>TAB<comments>

The last two fields are optional. A "line" may be made up of multiple actual lines in the file by using continuation lines, which are lines beginning with SPACE characters. The line breaks may be after TABS or in between addresses in a distribution list.

Nicknames (the first field) are short names that the user types instead of typing in the full address. There are several characters which aren't allowed in nicknames in order to avoid ambiguity when parsing the address (for example: spaces, commas, "@", ...).

The fullname field is usually stored as Last_name, First_name, in order that a sort on the fullname field comes out right. If there is a comma in the fullname, Pine will flip the first and last name around and get rid of the comma when using the entry in a composition. It isn't required that there be a comma, that's only useful if the user wants the entries to sort on last names.

The address field takes one of two forms, depending on whether the entry is a single (simple) address or a distribution list. For a simple entry, the address field is the email-address part of the address, i.e., the part that goes inside the brackets (<>). It is combined with the fullname field to form the complete address. For a distribution list, the <address> is in the format:

"(" <address>, <address>, <address>, ... ")"

Unlike the simple entry case, each of the addresses in a list can be a full RFC 822 address with fullname included, or it may be just the same as in the simple case. This way you can have a list which includes the fullnames of all the list members. In both the simple and list cases, addresses may also be other nicknames which appear in this address book or in one of the other address books. (Those nicknames are searched for by looking through the address books in the order they appear in the address book screen, with the first match winning.) Lists may be nested. If addresses refer to each other in a loop this is detected and flagged. The address will be changed to "**** address loop ****".

The optional fcc field is a folder name, just like the fcc field in the composer headers. If the first address in the To field of a composition comes from an address book entry with an fcc field, then that fcc is placed in the fcc header in the composer.

The comments field is just a free text field for storing comments about an entry. Neither the fcc nor the comments field is normally shown on the screen in the address book screen. You can only see them by Editing them. You may also search them with the WhereIs command.

The address book is displayed in the order that it is sorted in the file. When the user chooses a different sorting criterion, the file is actually sorted, not just the view of the file.

When the address book is written out, it is first written to a temporary file and if that write is successful it is renamed correctly. This guards against errors writing the file that might destroy the whole address book. The address book is re-written after each change.

The end-of-line character(s) in the address book file are those native to the system writing it. So it is <LF> on Unix and <CR><LF> on PC's. However, both Unix and PC versions of Pine can read either format, so it should be possible to share a read-only address book among the two populations (using NFS, for example). The end-of-line character for the LookUp file is always just <LF>, even on a PC. There is not currently any method built into Pine to access a remote address book (through IMAP or something like that). The only sharing possible is via some external remote file system or copying. It is very likely that a future version of Pine will be able to access remote address books using IMSP, when that becomes standardized and available.

Address Book Lookup File

Starting in 3.90 there is an additional file for each address book, called the LookUp file. It usually has the same name as the address book file with the suffix ".lu" appended. (It might have a different name if a file name length restriction prohibited that name.) This file is created and maintained by Pine. Its purpose is to speed up lookups for large address books and to reduce memory requirements for large address books. A fairly detailed description of how it is used is given in src/pine/adrbklib.h.

The lookup file changes whenever the address book itself is changed. If it doesn't exist, Pine attempts to create it. If Pine doesn't have permission to create the lookup file with the standard name, it will create a temporary version in a temp directory. You want to avoid this since it would have to be rebuilt every time Pine was run, and rebuilding takes a significant time for a large address book. So, if you're going to have a shared address book in a read-only directory, it is highly desirable to create the lookup file so that the users sharing it won't have to each create a copy in a temp directory. You can do that by running Pine and accessing the address book under a user id which does have permission to write the file (root, for example) or by using the *-create_lu* command line argument to Pine (as root, still). If users may be using a shared address book that needs updating, it is best to move the old address book to another name rather than copying over it. It is also best to make the lookup file for the new addrbook before moving it and the address book file into place, otherwise users may get stuck initializing the new file.

An effort is made to detect that an address book has been changed by another process. If a change is detected, the address book will be closed down and a new open will be attempted. If the new lookup file is in place when the open is tried, it will work smoothly. In normal operation (lookups and browsing the address book) the check to see if it has changed is just a heuristic to notice if things seem right. It isn't more rigorous because it needs to be fast. When a lookup is done, an offset into the address book is gotten from the LookUp file and a seek into the address book is done. It will check to see if the preceding character is an end-of-line character, which it should be. If it isn't, it figures it needs to rebuild the LookUp file. When an address book is about to be changed, a more fool-proof check is made. Several things in the file are checked to see that it is a LookUp file (magic number, size, ...) and that it is whole. Then, a timestamp in the LookUp file is compared to the mtime of the address book. If the timestamp is later than the mtime, everything is ok, otherwise, the address book has been changed and the new change is aborted.

The address book code has been completely rewritten for 3.90 and production experience with shared address books is nil at the time of this writing. We expect there may be some changes as experience is gained, and that some new tools may emerge (scripts to convert password files to shared address books, for example).

Checkpointing

Periodically Pine will save the whole mail folder to disk to prevent loss of any mail or mail status in the case that Pine gets interrupted, disconnected, or crashes. The period of time Pine waits to do the checkpoint is calculated to be minimally intrusive. The timing can be changed (but usually isn't) at compile time. Folder checkpointing happens for both local folders and those being accessed with IMAP. The delays are divided into three categories:

Good Time: This occurs when Pine has been idle for more than 30 seconds. In this case Pine will checkpoint if 12 changes to the file have been made or at least one change has been made and a checkpoint hasn't been done for five minutes.

Bad Time: This occurs just after Pine has executed some command. Pine will checkpoint if there are 36 outstanding changes to the mail file or at least one change and no checkpoint for ten minutes.

Very Bad Time: Done when composing a message. In this case, Pine will only checkpoint if at least 48 changes have been made or one change has been made in the last twenty minutes with no checkpoint.

Debug Files

If UNIX Pine is compiled with the compiler *DEBUG* option on (the default), then Pine will produce debugging output to a file. The file is normally *.pine-debugX* in the user's home directory where *X* goes from 1 to 4. Number 1 is always the most recent session and 4 the oldest. Four are saved because often the user has gone in and out of Pine a few times after a problem has occurred before the expert actually gets to look at it. The amount of output in the debug files varies with the debug level set when Pine is compiled and/or as a command line flag. The default is level 2. This shows very general things and records errors. Level 9 produces copious amounts of output for each keystroke.

PC-Pine creates a single debug file named *PINEDEBG.TXT* in the same directory as the *PINERC* file.

Filters

Pine is not designed to process email messages as they are delivered; rather Pine depends on the fact that some other program (sendmail, etc) will deliver messages and Pine simply reads the email folders which that "other" program creates. For this reason, Pine cannot filter incoming email into different folders. It can, however, work alongside most of the programs available over the Internet which perform this task. Pine is known to operate successfully with the Elm filter program and with Procmail.

Design changes introduced in Pine 3.8x facilitate Pine users filtering email. You still have to get a filtering program and configure it correctly, but Pine now allows users to specify a set of *incoming-folders*. Pine will separate out all the folders listed as *incoming-folders* and offer convenient access to these. We hope that in the future Pine will be able to offer new message counts for all of the incoming folders.

Folder Formats and Name Extensions

A folder is a group of messages. The default format used by Unix Pine is the Berkeley mail format. It is also used by the standard *mail* command and by *elm.* Unix Pine also understands message folders in other formats, such as Tenex, MH, MMDF, Carmel, and Netnews. (For more information about the carmel format, see the directory *./contrib/carmel* in the Pine distribution.)

PC-Pine reads and writes local (PC) folders in a special format similar to the Tenex format. Near as we can tell, PC-Pine is the only program to use this format. Beginning with version 3.90, PC-Pine includes a Read-Only driver for the Berkeley mailbox format in addition. That means that you can import Unix mail folders, or mount them via NFS or SMB, and PC-Pine can read them --but not modify them.

Extensions. In the past, file name extensions have been significant in both Unix Pine and PC-Pine, but this has caused more problems than it solved. Therefore, on Unix Pine extensions no longer have any special meaning, and this is the trend for PC-Pine as well.

By default, PC-Pine adds ".MTX" to the name of any local (PC) folders that are referenced, and suppresses the extension from the Folder List display. Now that PC-Pine can read more than one folder format, the MTX extension no longer implies a particular format, and is largely irrelevant. By using the "folder_extension" option, you can change this behavior. In particular, you may set "folder-extension" to the "null string" which tells PC-Pine to neither add nor hide-from-view *any* folder name extension.

The reason you might wish to over-ride the MTX default is that recent versions of PC-Pine have the ability to open (albeit READ-ONLY) normal Unix mail folders. Since it might be inconvenient to rename all of them to have an MTX extension, it is possible with this option to switch PC-Pine's behavior so that such folders can be seen and accessed without changing their names. However, doing this means that your existing PC-Pine local folders will have apparently changed their names. For example, if you had a local folder named "FOO" it will now appear in the Folder List as "FOO.MTX". If you wish to save additional messages to that folder, you will need to enter the full name, "FOO.MTX" at the Save prompt. Likewise for GOTO.

If you wish to permanently avoid having to deal with folder name extensions, you will need to set this option to the null string by entering two double- quote marks, and you will need to rename your existing local folders to not have an MTX extension. In DOS this can be done in one command, once you have changed to your mail directory: RENAME *.MTX *.

We don't know why you might wish to, but you could also use this option to tell PC-Pine to use an extension other than MTX. In this case, enter the three characters you desire to use in lieu of "MTX". Note that your existing folders will need to be renamed to correspond to this new extension.

Berkeley Mail Format

This format comes to us from the ancient UNIX mail program, */bin/mail.* (Note that this doesn't have anything to do with Berkeley, but we call it the Berkeley mail file format anyway.) This program was actually used to interactively read mail at one time, and is still used on many systems as the local delivery agent. In the Berkeley mail format, a folder is a simple text file. Each message (including the first) must start with a separator line which takes approximately the form:

```
From juser@u.example.edu  Wed Aug 11 14:32:33 1993
```

Each message ends with two blank lines. There are actually several different variations in the date part of the string, twenty at last count. Because of the format of the separators, lines in the mail message beginning with "From ", space included, risk being confused as message separator lines. Some mail programs will interpret any line beginning with "From " as a message separator, while others --including Pine-- will not be confused unless the line really looks like a message separator, complete with address and date. Such lines will be modified to begin with ">From ". In deference to other mail programs, you may also set the "save-will-quote-leading-froms" feature, in which case any line beginning with "From " will be modified as above. If you see this occasionally in incoming mail messages, the culprit is not Pine but the message delivery program being used at your site.

You can fool Pine into thinking a file is a mail folder by copying a suitable message separator from a real folder to the beginning of the file and wherever you want message boundaries. The vast majority of INBOXes Pine reads and folders it writes are of this format.

Tenex and MTX Formats

Like the Berkeley format, the Tenex folder format uses a single file per folder. Historically, the name of Tenex-format folders ended with *.txt*, but this rule is no longer enforced. The file format consists of a header line followed by the message text for each message. The header is in one of two forms:

```
dd-mmm-yy hh:mm:ss-zzz,n;ffffffffffff
dd-mmm-yyyy hh:mm:ss sssss,n;ffffffffffff
```

and is immediately followed by a newline (and the message text).

The fields in the formats are:

dd	two-digit day of month (leading space if a single-digit day)
mmm	three-letter English month name (Jan, Feb, etc.)
yy	two-digit year in 20th century (obsolete)
yyyy	four-digit year
hh	two-digit hour in 24-hour clock (leading zero if single-digit)
mm	two-digit minute (leading zero)
ss	two-digit second (leading zero)

zzz three-letter North American time zone (obsolete)
sssss signed four-digit international time zone as in RFC 822
n one or more digits of the size of the following message in bytes
ffffffffffff
twelve-digit octal flags value

Punctuation is as given above.

The time in the header is the time that message was written to the folder. The flags are interpreted as follows: the high order 30 bits are used to indicate user flags, the next two bits are reserved for future usage, the low four bits are used for system flags (010 = answered, 04 = flagged urgent, 02 = deleted, 01 = seen).

If a Tenex-format (or empty) file named *mail.txt* exists in a Pine user's home directory, this triggers special processing in Pine. When INBOX is opened, mail is automatically moved from */usr/spool/mail* into *mail.txt* in the user's home directory.

The format used by PC-Pine is identical to the Tenex format, with two exceptions: the folder name ends with *.MTX* instead of *.txt* (this is a requirement in the MTX format), and DOS-style CR/LF newlines are used instead of UNIX-style LF newlines.

Netnews Format

The netnews format is a read-only format which uses directories under /usr/spool/news as folders. The */usr/spool/news/* prefix is removed and all subsequent ''/'' (slash) characters are changed to ''.'' (period). For example, the netnews folder name *comp.mail.misc* refers to the directory name */usr/spool/news/comp/mail/misc*. In addition, the news folder name must appear in the file /usr/lib/news/active for it to be recognized. Individual messages are stored as files in that directory, with file names being the ASCII form of a number assigned to that message.

Folder Locking

There are two kinds of locking which Pine has to worry about. The first might be called program-contention locking. This affects the times when a program is performing actual updates on a folder. An update might be a message delivery program appending a message (*sendmail* delivering a message to an INBOX), status changes (checkpoints by Pine every few minutes) or deletion of messages (an expunge in Pine). For moderate sized mail messages, these operations should not last for more than a few seconds. The second kind of locking has to do with user-contention situations. This would be the case when one folder is shared by a group of people or even when one person starts multiple email sessions all of which access the same folders and INBOX.

There are two standard locking mechanisms which handle program-contention locking. To be on the safe side, Pine implements both of them. The older mechanism places a file *xxxx.lock* (where *xxxx* is the name of the file being locked) in the same directory as the file being locked. This makes use of the fact that directory operations are atomic in UNIX and mostly works across NFS. There are involved algorithms used to determine if a lock has been held for an excessive amount of time and should be broken. The second program-contention locking mechanism uses the *flock()* system call on the mailbox. This is much more efficient and the locks can't get stuck because they go away when the process that created them dies. This is usually found on 4BSD and related machines.

In addition to these, Pine—through the c-client library—provides robust locking which prevents several users (or several instances of the same user) having a mail file open (for update) at once. This user-contention lock is held the entire time that the folder is in use.

With IMAPd 7.3(63) and Pine 3.84 and higher, the second Pine session which attempts to open a particular folder (usually INBOX) with Pine will ''win''and That is to say, the second session will have read/write access to the folder. The first user's folder will become read-only. (Note that this is exactly the opposite of the behavior prior to Pine 3.84 where the second open was read-only. Having the latest open be read-write seems to match more closely with what users would like to have happen in this situation.) Pine's additional locking is only effective against multiple uses of Pine or other programs using the c-client library, such as *MailManager, ms, IMAPd* and a few others. Beginning with Pine 3.85, there is an *-o* command line flag to intentionally open a mailbox read-only.

Pine locking on UNIX systems works by creating lock files in */tmp* of the form *\usr\spool\mail\joe.* The system call *flock()* is then used on these files; the existence of the file alone does not constitute a lock. This lock is created when the folder is opened and destroyed when it is closed. When the folder is actually being written, the standard UNIX locks are also created.

If a folder is modified by some other program while Pine has it open, Pine will give up on that mail file, concluding it's best not to do any further reads or writes. This can happen if another mailer that doesn't observe Pine's user-contention locks (e.g. *elm* or *mail)* is run while Pine has the mail folder open. Pine checkpoints files every few minutes, so little data can be lost in these situations.

PC-Pine does not do any folder locking. It depends on IMAP servers to handle locking of remote folders. It is assumed that only one Pine session can be running on the PC at a time, so there is no contention issue around folders on the PC itself.

INBOX and Special Folders

The *INBOX* folder is treated specially. It is normally kept open constantly so that the arrival of new mail can be detected. The name *INBOX* refers to wherever new mail is retrieved on the system. If the *inbox-path* variable is set, then *INBOX* refers to that. IMAP servers understand the concept of *INBOX*, so specifying the folder {*imap.u.example.edu*}*INBOX* is meaningful. The case of the word INBOX is not important, but Pine tends to display it in all capital letters.

The folders for sent mail and saved messages folders are also somewhat special. They are automatically created if they are absent and recreated if they are deleted.

Internal Help Files

The file *pine.hlp* in the *pine* subdirectory of the distribution contains all the help text for Pine. On UNIX, it is compiled right into the Pine binary as strings. This is done to simplify installation and configuration. The *pine.hlp* file is in a special format that is documented at the beginning of the file. It is divided into sections, each with a name that winds up being referenced as a global variable. Some special formatting rules are used to keep things lined up and to allow for substitutions in the help text depending on whether the Pine session uses function keys or the standard alphabetic/mnemonic keys. This file is processed by two awk scripts and turned into C files that are compiled into Pine.

This scheme can increase efficiency because Pine can be compiled to have the strings as part of shared, read-only text. Rather than each process having to read in the help text from a file, the strings are shared by all executing processes on the machine and demand paged. This works on machines that have separate instruction and data space, but is only fully implemented in the NeXT (tested) and Dynix (not tested) ports.

PC-Pine, which tries to run on machines with as little as 640k of memory, leaves the Pine help text out of the executable. *PINE.EXE*, *PINE.HLP*, and *PINE.NDX* are all needed for PC-Pine's help system.

International Character Sets

While Pine was designed in the U.S. and used mostly for English-language correspondence, it is a goal for Pine to handle email in almost any language. Many sites outside of the U.S. run Pine in their native language. The default character set for Pine is US-ASCII. That can be changed in the personal or system-wide configuration file with the variable *character-set*.

When reading incoming email, Pine allows all character sets to pass through. Pine doesn't actually display the characters but simply passes them through; it is up to the actual display device to show the characters correctly. When composing email, Pine will accept input in any language and tag the message according to the *character-set* variable. Again, it is up to the input device to generate the correct sequences for the character set being used. The outgoing message is checked to see if it is all US-ASCII text (and contains no escape characters). In that case, the text will be labeled as US-ASCII even if the *character-set* variable is set to something else. The theory is that every reasonable character set will have US-ASCII as a subset, and that it makes sense to label the text with the lowest-common-denominator label so that more mailers will be able to display it.

The character sets are:

US-ASCII	Standard 7 bit English characters
ISO-8859-1	8 bit European "latin 1" character set
ISO-8859-2	8 bit European "latin 2" character set
ISO-8859-3	8 bit European "latin 3" character set
ISO-8859-4	8 bit European "latin 4" character set
ISO-8859-5	8 bit Latin and Cyrillic
ISO-8859-6	8 bit Latin and Arabic
ISO-8859-7	8 bit Latin and Greek
ISO-8859-8	8 bit Latin and Hebrew
ISO-8859-9	8 bit European "latin 5" character set
ISO-8859-10	8 bit European "latin 6" character set
KOI8-R	8 bit Latin and Russian
VISCII	8 bit Latin and Vietnamese
ISO-2022-JP	Latin and Japanese
ISO-2022-KR	Latin and Korean
UNICODE-1-1	Unicode
UNICODE-1-1-UTF-7	Mail-safe Unicode
ISO-2022-JP-2	Multilingual

In all of these except Japanese, the lower 7 bits are the same as US-ASCII. Even in

Japanese, the character set is the same as US-ASCII unless it has been shifted to an alternate interpretation.

Earlier versions of Pine made use of the character set tags associated with text in MIME to decide if the text should be displayed or not. Depending on the character set tag and the *character-set* variable in Pine, the text was either displayed as is, displayed with some characters filtered out, or not displayed at all. The current version uses a much simpler algorithm in order to maximize the chance that useful contents are readable by the user. It simply displays **all** messages of type text and makes no attempt to filter out characters that may be in the wrong character set. If the text is tagged as something other than US-ASCII and the tag does not match the character set that the *character-set* variable is set to, then a warning is printed at the start of the message. In that case, it is possible that the text will be displayed incorrectly. For example, if the text is one variant of ISO-8859 and the display device is another variant, some of the characters may show up on the screen as the wrong character. Or if the text is Japanese and the display device is not, some parts of the message may be total gibberish (which will look like ASCII gibberish). On the other hand, the parts of the Japanese message that really are US-ASCII will be readable in the midst of the gibberish.

In the case of PC-Pine, the character values cannot be passed through to the display device unaltered since MS-DOS uses various non-standard character sets called "Code Pages".

The mapping between DOS Code Page and standard character set is controlled by the "character-set" variable in the PINERC file and the PC's installed Code Page. PC-Pine will automatically map common characters in IBM Code Pages 437, 850, 860, 863, and 865 to ISO-8859-1 and back when the PINERC has "character-set=ISO-8859-1". Pine will also map common characters for IBM Code Page 866 to ISO-8859-5 and back when "character-set=ISO-8859-5". The mappings are bi-directional, and applied to all saved text attachments in the defined character set, messages exported, etc.

Alternatively, the translation tables can be configured externally and applied at run time whenever the "character-set=" variable is set to something other then "US-ASCII" (the default). PC-Pine looks in the text file pointed to by the environment variable "ISO_TO_CP" for the table to use for mapping text matching the type defined by the "character-set=" variable into the local Code Page value. PC-Pine looks in the text file pointed to by the environment variable "CP_TO_ISO" for the table to use for mapping text in the local Code Page into outbound text tagged with the "character-set=" variable's value.

A text file containing a character set mapping table is expected to contain 256 elements where each element is a decimal number separated from the next element by white-space (space, tab or newline, but no commas!). The index of the element is the character's value in the source character set, and the element's value is the corresponding character's value in the destination character set.

Interrupted and Postponed Messages

If the user is composing mail and is interrupted by being disconnected (SIGHUP, SIGTERM or end of file on the standard input), Pine will save the interrupted composition and allow the user to continue it when he or she resumes Pine. As the next Pine session starts, a message will be given that an interrupted message can be continued. To continue the interrupted message, simply go into the composer. To get rid of the interrupted message, go into the composer and then cancel the message with *^C*.

Composition of half-done messages may be postponed to a later time by giving the *ˆO* command. Other messages can be composed while postponed messages wait. All of the postponed messages are kept in a single folder. Postponing is a good way to quickly reference other messages while composing.

Message Status

The c-client library allows for several flags or status marks to be set for each message. Pine uses four of these flags: UNSEEN, DELETED, ANSWERED, and FLAGGED. The "N" in Pine's FOLDER INDEX means that a message is unseen–it has not been read from this folder yet. The "D" means that a message is marked for deletion. Messages marked with "D" are removed when the user *expunges* the folder (which usually happens when the folder is closed or the user quits Pine). The "A" in Pine's FOLDER INDEX means that the message has been replied-to. The "*" in Pine's FOLDER INDEX means that the message has been ''flagged'' as important. That is, the user used the *Flag* command to turn the FLAGGED flag on. This flag can mean whatever the user wants it to mean. It is just a way to mark some messages as being different from others. It will usually probably be used to mark a message as somehow being ''important''. For Berkeley format folders, the message status is written into the email folder itself on the header lines marked *Status:* and *X-Status*. In Tenex and PC-Pine's MTX folder formats, the status goes into the 36-bit octal flags.

MIME–Reading a Message

Pine should be able to handle just about any MIME message. When a MIME message is received, Pine will display a list of all the parts, their types and sizes. It will display the attachments when possible and appropriate and allow users to save all other attachments.

Starting with version 3.90, Pine honors the "mailcap" configuration system for specifying external programs for handling attachments. The mailcap file maps MIME attachment types to the external programs loaded on your system which can display and/or print the file. A sample mailcap file comes bundled with the Pine distribution. It includes comments which explain the syntax you need to use for mailcap. With the mailcap file, any program (mail readers, newsreaders, WWW clients) can use the same configuration for handling MIME-encoded data.

If a $MAILCAPS environment variable is defined, Pine will use that to look for one or more mailcap files, which are combined. In the absence of $MAILCAPS, Unix Pine will look for a personal mailcap file in ˜/.mailcap and combine that with a system-wide file in /etc/mailcap. PC-Pine will look for a file named *MAILCAP* in the same directory as the *PINERC* file, and/or the directory containing the *PINE.EXE* executable.

Messages which include *rich text* or *enriched text* in the main body will be displayed in a very limited way (it will show bold and underlining).

If Pine sees a MIME message part tagged as type IMAGE, and Pine's *image-viewer*. configuration variable is set, Pine will attempt to send that attachment to the named image viewing program. In the case of UNIX Pine, the DISPLAY environment variable is checked to see if an X-terminal is being used (which can handle the images). If the *image-viewer* variable is not set, Pine uses the *mailcap* system to determine what to do with IMAGE types, just as it does for any other non-TEXT type, e.g. type APPLICATION. For MIME's generic "catch all" type, APPLICATION/OCTET-STREAM, the *mailcap* file will probably not specify any action, but Pine users may always Save any MIME attachment to a file.

If an attachment is just text (tagged with "text/plain" in the MIME header), then Pine will use an internal viewer module to display the attachment. International character sets in attachments are handled in the same way as they are in regular email messages. Some text attachments, specifically those which are just other email messages forwarded as MIME messages, are displayed as part of the main body of the message. This distinction allows easy display when possible (the forward as MIME case) and use of an attachment viewer when that is desirable (the plain text file attachment case).

If the parts of a multipart message are alternate versions of the same thing Pine will select and display the one best suited. For parts of type "message/external-body", the parameters showing the retrieval method will be displayed, but the retrieval process is not yet automated. Messages of type "message/partial" are not currently supported.

MIME–Sending a Message

There are two important factors when trying to include an attachment in a message: encoding and labeling. Pine has rules for both of these which try to assure that the message goes out in a form that is robust and can be handled by other MIME mail readers.

MIME has two ways of encoding data–Quoted-Printable and Base64. Quoted-Printable leaves the ASCII text alone and only changes 8-bit characters to "=" followed by the hex digits. For example, "=09" is a tab. It has the advantage that it is mostly readable and that it allows for end of line conversions between unlike systems. Base64 encoding is similar to *uuencode* or *btoa* and just encodes a raw bit stream. This encoding is designed to get text and binary files through even the most improperly implemented and configured gateways intact, even those that distort uuencoded data.

All attachments are encoded using Base64 encoding. This is so that the attachment will arrive at the other end looking exactly like it did when it was sent. Since Base64 is completely unreadable except by MIME-capable mailers or programs, there is an obvious tradeoff being made here. We chose to ensure absolutely reliable transport of attachments at the cost of requiring a MIME-capable mailer to read them. If the user doesn't want absolute integrity he or she may always *include* text (with the *ˆR* command) in the body of a message instead of attaching it. With this policy, the only time quoted-printable encoding is used is when the main body of a message includes special foreign language characters.

When an attachment is to be sent, Pine sniffs through it to try to set the right label (content-type and subtype). An attachment with any lines longer than 500 characters in it or more than 10% of the characters are 8-bit it will be considered binary data. Pine will recognize (and correctly label) a few special types including GIF, JPEG, PostScript, and some audio formats.

If it is not binary data (has only a small proportion of 8-bit characters in it,) the attachment is considered 8-bit text. 8-bit text attachments are labeled "text/plain" with charset set to the value of the user's *character-set* variable. If an attachment is ASCII (no 8-bit characters) and contains no *ESCAPE, ˆN,* or *ˆO* characters (the characters used by some international character sets), then it is considered plain ASCII text. Such attachments are given the MIME label "text/plain; charset=US-ASCII", regardless of the setting of the user's *character-set* variable.

All other attachments are unrecognized and therefore given the generic MIME label "application/octet-stream".

New Mail Notification

Pine checks for new mail in the *INBOX* and in the currently open folder at least every two and a half minutes. It used to be 30 seconds instead of 150 seconds, but we increased it in order to reduce the load on large systems with lots of Pine users. The value can be changed at compile-time in the pine/os.h file. If you really don't want to wait you can force a new mail check by pressing *N Next* with the cursor on the last message of the message index or by redrawing the screen with a *^L*.

When there is new mail, the message(s) will appear in the index, the screen will beep, and a notice showing the sender and subject will be displayed. If there has been more than one new message since you last issued a command to Pine, the notice will show the count of new messages and the sender of the most recent one.

Questions have arisen about the interaction between Pine and external mail notification routines (biff, csh, login). Firstly and unfortunately, we have found no PC based program that will check for email on an IMAP server when PC-Pine is not running. If you find one, please tell us.

The UNIX case is more complicated. Pine sets the modification and access time on a file every time it performs a write operation (status change or expunge). You need to see which of these your email notification program is looking at to know how it will behave with Pine.

NFS

It is possible to access mail folders on *NFS* mounted volumes with Pine, but there are some drawbacks to doing this, especially in the case of incoming-message folders that may be concurrently updated by Pine and the system's mail delivery agent. One concern is that Pine's user-contention locks don't work because */tmp* is usually not shared, and even if it was, *flock()* doesn't work across *NFS.*

The implementation of the standard UNIX ".lock" file locking has been modified to work with *NFS* as follows. Standard hitching post locking is used so first a uniquely named file is created, usually something like *xxxx.host.time.pid.* Then a link to it is created named *xxxx.lock* where the folder being locked is *xxxx.* This file constitutes the lock. This is a standard UNIX locking scheme. After the link returns, a *stat(2)* is done on the file. If the file has two links, it is concluded that the lock succeeded and it is safe to proceed.

In order to minimize the risks of locking failures via NFS, we strongly recommend using IMAP rather than NFS to access remote incoming message folders, e.g. your INBOX. However, it is generally safe to access personal saved-message folders via *NFS* since it is unlikely that more than one process will be updating those folders at any given time. Still, some problems may occur when two Pine sessions try to access the same mail folder from different hosts without using IMAP. Imagine the scenario: Pine-A performs a write that changes the folder. Pine-B then attempts to perform a write on the same folder. Pine-B will get upset that the file has been changed from underneath it and abort operations on the folder. Pine-B will continue to display mail from the folder that it has in its internal cache, but it will not read or write any further data. The only thing that will be lost out of the Pine-B session when this happens is the last few status changes.

If other mail readers besides Pine are involved, all bets are off. Typically, mailers don't take any precautions against a user opening a mailbox more than once and no special precautions are taken to prevent *NFS* problems.

Printers and Printing

UNIX Pine can print to the standard UNIX line printers or to generic printers attached to ANSI terminals using the escape sequences to turn the printer on and off. The user has a choice of three printers in the configuration.

The first setting, *attached-to-ansi,* makes use of escape sequences on ANSI/VT100 terminals. It uses "<ESC>[5i" to begin directing all output sent to the terminal to the printer and then "<ESC>[6i" to return to normal. Pine will send these escape sequences if the printer is set to *attached-to-ansi.* This works with most ANSI/VT100 emulators on Macs and PCs such as kermit, NCSA telnet, VersaTerm Pro, and WinQVT. Various terminal emulators implement the print feature differently. For example, NCSA telnet requires "capfile = PRN" in the *config.tel* file. Attached-to-ansi printing doesn't work at all with the telnet provided with PC-NFS.

The second selection is the standard UNIX print command. The default is *lpr*, but it can be changed on a system basis to anything so desired in */usr/local/lib/pine.conf.*

The third selection is the user's personal choice for a UNIX print command. The text to be printed is piped into the command. *Enscript* or *lpr* with options are popular choices. The actual command is retained even if one of the other print selections is used for a while.

If you have a PostScript printer attached to a PC or Macintosh, then you will need to use a utility called *ansiprt* to get printouts on your printer. *Ansiprt* source code and details can be found in the *./contrib* directory of the Pine distribution.

The three printer choices are for UNIX Pine only. PC-Pine for DOS can only print to the locally attached printer. All printing on PC-Pine (DOS) is done via ROM BIOS Print Services (Int 17h). After verifying the existence of a local printer via the BIOS Equipment-List Service (Int 11h), it simply sends the message text, character by character, to the first printer found using ASCII CR and LF at the end of lines and followed by an ASCII FF. Note, some system adjustments using the PC's "MODE" command may be required if the printer is not on the first parallel port. PC-Pine cannot generate PostScript, so printing to exclusively PostScript printers does not work.

PC-Pine for Winsock uses the MS-Windows printer interface. A Pine print command will bring up a standard MS-Windows printer dialog box.

Save and Export

Pine users get two options for moving messages in Pine: *save* and *export*. Save is used when the message should remain ‘‘in the Pine realm.’’ Saved messages include the complete header (including header lines normally hidden by Pine), are placed in a Pine folder collection and accumulate in a standard folder format which Pine can read. In contrast, the *export* command is used to write the contents of a message to a file for use outside of Pine. Messages which have been exported are placed in the user's home directory (unless the feature *use-current-dir* is turned on), not in a Pine folder collection. Unless FullHeaderMode is toggled on, all delivery-oriented headers are stripped from the message. Even with *export,* Pine retains message separators so that multiple messages can accumulate in a single file and subsequently be accessed as a folder. On UNIX systems, the *export* command pays attention to the standard *umask* for the setting of the file permissions.

Sent Mail

Pine's default behavior is to keep a copy of each outgoing message in a special "sent mail" folder. This folder is also called the fcc for "file carbon copy". The existence, location and name of the sent mail folder are all configurable. Sent mail archiving can be turned off by setting the configuration variable *default-fcc*="". The sent mail folder is assumed to be in the default collection for saves, which is the first collection named in *folder-collections*. The name of the folder can be chosen by entering a name in *default-fcc*. With PC-Pine, this can be a bit complicated. If the default collection for saves is local (DOS), then the *default-fcc* needs to be "SENTMAIL", which is syntax for a DOS file. However, if the default collection for saves is remote, then the *default-fcc* needs to be "sent-mail" to match the UNIX syntax.

The configuration variable *fcc-name-rule* also plays a role in selecting the folder to save sent mail in. See the documentation on it in the section on configuration variables.

The danger here is that the sent mail could grow without bound. For this reason, we thought it useful to encourage the users to periodically prune their sent mail folder. The first time Pine is used each month it will offer to archive all messages sent from the month before. Pine also offers to delete all the sent mail archive folders which are more than 1 month old. If the user or system has disabled sent mail archiving (by setting the configuration variable *default-fcc*="") or if the fcc folder is a remote/IMAP folder then there will be no pruning question.

It is likely that Pine will be improved so that users can set the time increment for pruning (weekly, monthly, yearly, never) but that has not been implemented yet.

Spell Checker

Spell checking is available for UNIX Pine only. We could not find an appropriate PC based spell checker to hook into PC-Pine. Even UNIX Pine depends on the system for its spell checking and dictionary. Pico, the text editor, uses the same spell checking scheme as Pine.

Lines beginning with ">" (usually messages included in replies) are not checked. The message text to be checked is on the standard input and the incorrect words are expected on the standard output.

The default spell checker is UNIX *spell*. You can replace this at compile time for the whole system. Pine also respects the environment variable *SPELL*. If it is set, Pine will use that as the spelling checker. The spelling checker reads its words from a standard dictionary on the system. Below is a description, contributed by Bob Hurt, of how you can create your own personal dictionary with additional ‘‘correct’’ words.

Step 1:

Make a file with all the words you want to include in your new dictionary. I did mine with one word per line in alphabetical order. Caps don’t matter at all, as far as I know.

Step 2:

At the UNIX prompt, type "cat [word file] | spellin /usr/dict/hlista > [new dict name]" where [word file] is the file you just created and [new dict name] is the name of the new dictionary that Pine will look at instead of the standard */usr/dict/hlista.* I named my word file *.bobwords* and my dictionary *.bobspell* so I don’t have to see them when I do a *ls* command (*ls* doesn’t list "dot" files). I also put the above command into my *.alias* file as the command *makedict* so I can add a word to my word file and easily recreate my dictionary. NOTE: the new dictionary is in something called a "hashed" format, and can’t be read normally.

Step 3:

Check your new dictionary. At the UNIX prompt, type "cat [word file] | spellout [new dict name]" If you did everything correctly, it should just give you another prompt. If it lists any of the words in your file, something is wrong. I can try to help if all else fails.

Step 4:

Now you have to tell UNIX to use your dictionary instead of the standard one by setting the environment variable *SPELL* to access your dictionary. Go into your *.login* or *.cshrc* file in your home directory (it doesn't seem to make a difference which one you use) and add the line

setenv SPELL "spell -d [new dict name]"

I also created an alias for *SPELL* in my *.alias* file so I can use the UNIX *spell* command to spell-check a file outside of Pine. (The *.alias* line is: alias spell 'spell -d [new dict name]')

Step 5:

Now you need to logoff and log back on to let UNIX look at your *.login* (or *.cshrc*) file.

Here is an alternative method suggested by Zachary Leber:

Create a list (e.g. *.zachwords*) with the upper case followed by lower case words, sorted alphabetically.

Add this line to *.cshrc*:

setenv SPELL 'spell +/home/ie/rsa/.zachwords'

The limitation here is that the path must be absolute (e.g. +˜/*.zachwords* doesn't work).

My man pages for spell show this + flag to be an easy way to do the exception list. This way you don't have to bother with hash lists or rehashing, and it seems to work across several platforms.

Terminal Emulation and Key Mapping

Pine has been designed to require as little as possible from the terminal. At the minimum, Pine requires cursor positioning, clear to end of line, and inverse video. Unfortunately, there are terminals that are missing some of these such as a vt52. Pine makes no assumptions as to whether the terminal wraps or doesn't wrap. If the terminal has other capabilities it may use some of them. Pine won't run well on older terminals that require a space on the screen to change video attributes, such as the Televideo 925. One can get around this on some terminals by using "protected field" mode. The terminal can be made to go into protected mode for reverse video, and then reverse video is assigned to protected mode.

Pine handles screens of most any size and resizing on the fly. It catches SIGWINCH and does the appropriate thing. A screen one line high will display only the new mail notification. Screens that are less than ten columns wide don't format very nicely or work well, but will function fine again once resized to something large. Pine sets an internal maximum screen size (currently 170x200) and decides to use either *termcap* or *terminfo* when it is compiled.

On the input side of things, Pine uses all the standard keys, most of the control keys and (in function-key mode) the function keys. Pine avoids certain control keys, specifically ˆS, ˆQ, ˆH, and ˆ\ because they have other meanings outside of Pine (they control data flow, etc.) *ˆH* is treated the same as the *delete* key, so the *backspace* or *delete* keys always works regardless of any configuration. In an upcoming version, there will be an option to have the *delete* key behave like ˆD rather than ˆH.

Sometimes a communications program or communications server in between you and the other end will eat certain control characters. There is a work-around when you need it. If you type two escape characters followed by a character that will be interpreted as the character with the control key depressed. For example, *ESC ESC T* is equivalent to *ˆT*.

When a function key is pressed and Pine is in regular (non-function key) mode, Pine traps escape sequences for a number of common function keys so users don't get an error message or have an unexpected command executed for each character in the function key's escape sequence. Pine expects the following escape sequences from terminals defined as VT100:

	ANSI/VT100
F1:	<ESC>OP
F2:	<ESC>OQ
F3:	<ESC>OR
F4:	<ESC>OS
F5:	<ESC>Op
F6:	<ESC>Oq
F7:	<ESC>Or
F8:	<ESC>Os
F9:	<ESC>Ot
F10:	<ESC>Ou
F11:	<ESC>Ov

Arrow keys are a special case. Pine has the escape sequences for a number of conventions for arrow keys hard coded and does not use *termcap* to discover them. This is because *termcap* is sometimes incorrect, and because many users have PC's running terminal emulators that don't conform exactly to what they claim to emulate. Some arrow keys on old terminals send single control characters like *ˆK* (one even sends ˆ\). These arrow keys will not work with Pine. The most popular escape sequences for arrow keys are:

Up:	<ESC>[A	<ESC>?x	<ESC>A	<ESC>OA
Down:	<ESC>[B	<ESC>?r	<ESC>B	<ESC>OB
Right:	<ESC>[C	<ESC>?v	<ESC>C	<ESC>OC
Left:	<ESC>[D	<ESC>?t	<ESC>D	<ESC>OD

It is possible to configure an NCD X-terminal so that some of the special keys operate. Brad Greer contributes these instructions:

1. In your *.Xdefaults* file, include the following "translations", using lower hex values:

```
Pine*VT100.Translations: #override \n\
<Key>Delete:    string(0x04)    \n\
<Key>End:       string(0x05)    \n\
<Key>Escape:    string(0x03)    \n\
<Key>Home:      string(0x01)    \n\
```

```
<Key>Next:      string(0x16)   \n\
<Key>Prior:     string(0x19)   \n\
<Key>KP_Enter:  string(0x18)   \n\
```

2. Start up Pine from an *xterm*, and specify a "resource name". This resource name will allow the user to specify resources for Pine (that deviate from the defaults). For example, *xterm -name Pine -e pine &* (the resource name *Pine* corresponds to the translations just added in the *.Xdefaults* file).

Section 7–Notes for Porting and Modification

Porting Pine to Other Platforms

Substantial effort has gone into making Pine/Pico portable. There are still, of course, a number of machine dependencies. Some of the ports are well-tested and some are untested. In particular, the most heavily used ports are the Ultrix, NeXT, DOS, and PTX ports.

Each platform is given a three letter name (see the file *doc/pine-ports*). Make up a new one for your new port. We've attempted to bring all potential platform dependencies into three files: *os-xxx.h, os-xxx.c,* and *makefile.xxx* where *xxx* is the three letter name of the port. Thus any new port will hopefully just result in new versions of these files and some notes for the *pine-ports* file. There are actually nine new files needed, because there is a set of these files in the c-client, Pico, and Pine source directories. (As you can tell by reading this technical note, Pine originated on Unix systems. There are still probably many Unix dependencies built in, but these should be diminishing now that there are *DOS, Windows,* and *VMS* ports. Regrettably, the source code is full of instances of "ifdef DOS". Most of these are due to memory limit problems on *DOS* as opposed to actual system dependencies.

The makefiles are kept as simple and straight-forward as possible, because many previous attempts at automatically figuring out what to do seem to have become complex and ineffective in what they set out to do: which is to make compiling and installing the program easy. Each port is for a specific hardware/software platform, also because past attempts to generalize on versions of Unix or some CPU architecture don't seem to have gained much. Thus, there is a separate makefile for each platform that calls the appropriate compiler and linker with the appropriate flags. Most of these makefiles are pretty similar. The makefile also specifies which of the *os-xxx.c* and *os-xxx.h* files to use. It is the root from which all platform dependencies are selected. In most cases the makefile also defines a symbol named after the platform on which there can be dependencies in the source code, though we've tried to minimize relying on this where reasonable. Pine, Pico, and the C-client don't quite do everything the same (there are at least three separate authors involved). Basically, to build the source in one of the directories, run *make -f makefile.xxx* where *xxx* is the three-letter name of the platform. That's all the *build* script does. When starting a new port in the *pine* directory, there is a generic makefile called *makefile.gen* which should be a good starting point.

The file *os-xxx.h* is used for general platform dependent #*include*'s and #*defines*. In the *pine* directory these *.h* files are located in the *osdep* subdirectory. All the include files that have been found to vary from one platform to another are also included here. In the case of Pico, there is only one *os-xxx.h* file called *os-unx.h* for most of the supported Unix ports and inside it are #*ifdefs* based on the platform specific symbol defined in the makefile. On the other hand, Pine now has a separate *os-xxx.h* file for each platform. There are a number of Pine configuration settings that are defined here, as well, such as the place it looks for certain files, defaults for the printer and folder names, the maximum screen size, and so on. For the Pine portion of the port, start by looking at the generic *os-gen.h* file and comparing it to some of the specific *os-xxx.h* files in *osdep.*

The *os-xxx.c* file contains functions that are potentially platform dependent. Again, the idea is to gather all the dependencies in one place. Pico uses the same strategy here as it uses with *os-unx.h.* That is, there is a single *os-unx.c* file for most of the Unix ports. Pine uses a complicated looking method to produce the *os-xxx.c* file from a set of included files. Each included file usually contains a single function and we've found that there are usually only a couple different implementations of each function in the ports we've done so far. Hopefully, coming up with an *os-xxx.c* for a new port will usually be a matter of including the right set of these already written functions. This is done by writing a new *os-xxx.ic* file in the *osdep* subdirectory. Start with the generic *os-gen.ic*, as you did with the *os-gen.h* file above.

We strongly encourage that no changes be made to the general source when porting and that all changes be contained in the three/nine system dependent files if possible. The object is to maintain source code integrity and assimilate ports to new platforms rapidly. The more conventional way to do this is with a large collection of *#ifdefs*. The problem with this is that adding a port for a new platform implies changing the source code for all the other platforms and thereby risks breaking them. (We readily admit that there are still too many *ifdefs* in the code, but we haven't had time to devote to fully cleaning that up.)

If you do port Pine to a new platform we hope that you will send us the changes required so that we may attempt to include it in a later release. Thanks!

Test Checklist

The following is a checklist of some things to check when testing a new port:

___ Sending mail, check that headers are correct
___ Sending mail with attachments
___ Sending mail with SMTP server
___ Sending mail without SMTP server
___ Sending mail with list of two SMTP servers, first one doesn't answer
___ Replying to and forwarding a message
___ Postponing messages under composition
___ Composer operations
___ Alternate editor, enable-alternate-editor-implicitly
___ Make sure local user names are expanded
___ Test spelling checker
___ Catching of SIGHUP while message is being composed
___ Setting of variables in *.pinerc*
___ New mail notification. Should happen with Pine idle to check timeouts
___ Reading mail (attachments, MIME, MIME with mailcap viewers)
___ Deleting, undeleting, expunging, sorting
___ Expunge to empty folder
___ Make sure that ~ expansion works in config files
___ Make sure that $VAR expansion works in config files
___ Save message to folder, check error conditions such as permission denied
___ Export message with FullHeaderMode on and off
___ Checkpointing (see the section on checkpointing)
___ Open IMAP and RIMAP folders
___ Default-fcc on remote IMAP server
___ Fcc-name-rule, fcc in addrbook (while composing)
___ Test opening bogus folders: invalid format, no permission
___ Open a USENET news group, list in folder-lister, read news, post news
___ Command line arguments
___ Change password
___ Lock keyboard
___ Address book operations (edit, delete, add, lists, whereis, composeto)
___ ReadOnly address book

___ Look at addrbook, change addrbook-sort-rule in Config, go back to addrbook screen
___ No permission to write in same directory as addrbook, should create addrbook.lu in a temp directory
___ Multiple address books
___ Address book loops from one addrbook to another and back
___ TakeAddr command with one address, with multiple addresses
___ TakeAddr command with ReadOnly address books
___ TakeAddr command with one of two address books ReadOnly
___ Send mail with empty address book
___ Config Screen operation, does pinerc get written?
___ Make sure SIGTSTP, ^Z works
___ Pinef
___ Sent-mail pruning (set back *last-time-prune-questioned* variable)
___ Printing using all three printer configurations, various screens
___ View help text and news
___ Folder list operations (rename, create, delete...)
___ Saved-msg-name-rule
___ Screen redrawing in various screens (^L)
___ Window resizing in various screens
___ Error messages for incorrect terminal types (try "foo" and "vt52")
___ Reading of /*usr*/*local*/*lib*/*pine.conf*
___ Fixing variables and features in /*usr*/*local*/*lib*/*pine.conf.fixed*
___ Flag command (check message status changed in mail folder)
___ Initial-keystroke-list
___ Aggregate operations (save, delete, export, takeaddr, ...)
___ Build xxx from scratch, build clean

- Pine Technical Notes -

Version 3.90, August 1994