Pine Technical Notes

Version 3.0, July 1992

Section I -- Installation, Design and Internals

Building and Installation

Hopefully, by this time you've already built Pine and tried it out. If not, you should be able to do it without too much trouble by picking the platform you want to build it for and giving the command "./build xxx" where "xxx" is the three letter abbreviation for the platform name. The list of these abbreviations is in the file "doc/pine-ports". You must be in the root of the source tree when you give the build command. If there's nothing in pine-ports for the system you have, building Pine and Pico will be a little or a lot more involved. The first thing to try is to pick a platform you think is close and try it, for example a different version of the same operating system. If it doesn't work you can try another. Giving the command "./build clean" between attempts is a good idea. It makes sure you start the compilation from scratch each time. If nothing works then some real porting will have to be done. See the sections on bugs, porting and source code below. The source code is about 4Mb and fully compiled it takes up 7-10Mb depending on the platform. You can do it in less space if you build Pine and Pico by hand and delete .o files as you compile each library.

When you're done building Pine you should have four binaries in the bin directory. If they didn't all compile you may still be OK. The binary, imapd, is only needed if you want to operate pine in client-server mode and mtest is used mostly for testing imapd. The Pine binary is the mailer and the Pico binary is a stand alone text editor that's similar to the message composer. Pico is not needed to run Pine or vice versa.

Usually, all you need to do to install Pine and Pico is to copy the pine and pico binaries from the bin directory to the local system bin directory so they can be execute by all users. /usr/local/bin is most frequently used. /usr/bin is also a possible place but they will get mixed in with your standard system programs if you put them there. All the help text is compiled into Pine so there are no required auxiliary files.

There are however, two optional auxiliary files: /usr/local/lib/pine.info and /usr/local/lib/pine.conf. The first one is can contain 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 suitably generic text is shown (that suggests sending mail to root as one way to get help). The other file is used to configure the 2 0 or so options that Pine has. See the section on configuration in this document.

Here's a step by step recipe:

1. Figure out what platform you're building Pine for. You can give the command "./build help" to get a short list of platforms. If it's not in that list you can look in the file doc/pine-ports (give the command "more doc/pine-ports" to view it). What you need is the three letter code for the platform. Some examples are "nxt" for a NeXT cubes and slabs and "ult" for Ultrix.

2. Make sure your in the root of the pine source. When you type "ls" you should see the following files and directories:

README	build	contrib	imapd	pico
bin	c-client	doc	makefile	pine

3. You can make sure you're getting a clean start by giving the command "./build clean". This should take 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 10 minutes or so. You shouldn't see any compiler warnings or errors, though there might be some if your system doesn't match the system you've specified to build.

5. When the compilation is complete the sizes of the four binaries built will be displayed. The binaries are in the source directories with links to the binaries are in the "bin" directories. You can just get them out of "bin".

Design goals

Pine was designed for novice computer users. To achieve this, features included were limited and carefully selected. This may have made Pine less attractive for power users, but the assumption was that there were many power user e-mail tools available for UNIX systems. It seemed very important to carefully consider questions that users asked. Advanced users are articulate about the changes they would like. Beginning users are not articulate and may walk away confused, not commenting if a program doesn't work out for them. Thus, if one continues development of a program based on users comments one may wind up with something suitable for advanced users, but lose the beginning users. To keep to the design goals, the questions users asked about Pine were examined to determine the areas that weren't clear. That is, when a user asked a question it was assumed that Pine could be clearer about the way it worked rather than assuming the user didn't understand.

Some other principles are:

- The underlying model presented to the user has to be simple and clear. To this end a lot of UNIX is hidden. For example, there is a simple list of folders without hierarchy and pathnames.

- It's better to have a few general, easily understood, commands that can be repeated than to have some more sophisticated command that will do the job all at once. This is one reason Pine has no aggregate operations.

- When ever the user has to select a command, file name, address etc, the user is given a menu or can get a menu to make the selection from. If possible the menu is as complete as possible (no hidden commands), small and well thought out. This isn't entirely possible with things like addresses, but the address book helps.

- Pine provides immediate feed back for the user with each operation.

- The system should be very tolerant of user errors. It should warn him if he's about do to something irreversible or provide a way to undo something. This way the user can learn by exploration with out 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. (At the moment there isn't any manual anyway).

- The size of the system should be kept small so the user doesn't feel like there are all these commands and concepts that he **should** know while using the system.

IMAP (The Interactive Mail Access Protocol)

Pine is capable of operating as an IMAP client. The syntax for naming an IMAP mailbox is {hostname}mailbox. Hostname is a standard DNS hostname and mailbox is the path of the mail file to be opened. It may be "inbox" to specify the systems idea of the inbox, or it may be a file path name relative to the users home directory. For example "mail/sent-mail". The remote host must be running an imapd and the user will be required to authenticate himself with a user name and password.

If "rimap" authentication is enabled on the remote host an .rhosts file on the server will allow the user to bypass typing the password. This works by having the client use the standard rsh mechanism to connect to the server which results in two extra rsh processes on the client. Basically what is happening here 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. Enabling "rimap" authentication is done by creating a link called rimapd to imapd.

Installing imapd requires placing the binary in the appropriate directory, usually /usr/etc and adding entries to /etc/services and /etc/inetd.conf. The following line is appropriate for /etc/services:

```
imap        143/tcp        # Mail transfer
```

and this 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 path name in /etc/inetd.conf must match the path where imapd is installed.

At this point one may ask, what is IMAP and why is it useful? A complete answer would be fairly long. Basically IMAP is a mail access protocol as distinguished from a mail transfer protocol. It's intended for interactive mail access. It's closest kin is POP, the Post Office Protocol, but it exceeds POP in many ways. POP works by transferring an entire mailbox to the client where all the mail is kept. IMAP retains all mail on the server and allows the client to retrieve mail as it's read and then send commands back to the server to specify what is to be done with the mail. The advantages of this is that it works well over low bandwidth lines because only what's needed is transferred, then mail is cached on the client. However, POP is appropriate where connect time charges are high. Other advantages of IMAP are that the mail is stored on a reliable, always up, server; that the mail can be accessed from many different clients in different places and is not stuck on a PC (probably with no backups); and that it allows one to take advantage of cheaper CPU cycles on the desk top to drive the user interface rather than expensive cycles on time sharing machines. One imap client can also be used to access several mail servers at once. The general idea is to run a mail client on the machine where a user has his home directory, to facilitate moving information between the mail system and user's file system, but to deliver mail to and maintain one's inbox on a high availability server.

Currently, Pine supports having one's inbox remote via IMAP. This feature is invoked by entering a {host}inbox value for the inbox-path option either the global pine configuration file or in a users .pinerc file. Access to other remote folders require the users to type the full IMAP mail box specification. Future plans include simpler access to remote folders other than inbox.

Some other sources of information are RFC-1176 and imap-viewgraphs.ps included here. You can also fetch the whole imap source tree and developers kit from ftp.cac.washington.edu. The IMAP protocol and the c-client library that implements it are authored by Mark Crispin.

Address Formats (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 "address (full name)" format. The intent of the standard is that items in parenthesis should only be for comments. Pine displays and generates the newer format, but will parse the old format and turn it into the new one.

Pine also fully qualifies all addresses on outgoing mail with the host or domain name as described in the section on domain names. This makes addresses more clear and unambiguous as networks get larger and span greater distances and gives a hint to the user that the network extends beyond the local organization. Typing the fully qualified address is not required, just the user name is sufficient. When the user moves the cursor out of the header field the address will be expanded and fully qualified. Note that because the newer format is used, commas are required to separate addresses since the full name part may have spaces in it. If any special characters as defined in RFC-822 appear in the full name, quotes are required around the address. Pine will insert them automatically. The common cases where this happens are with periods after initials and parentheses. A good substitute for parentheses are curly braces. They are not considered special.

On some UUCP connected machines and for UUCP addresses, fully qualified addresses may seem to confuse or break things. This is probably not the case though, because the local hostname is what is added. The mailer will recognize it as the local hostname and drop that part first before going on to process the rest of the address.

Pine also 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 index.

Known Bugs and Problems

In general, see the to-do-xxxx file included with this documentation. At the U.W. Pine is used by 3,000 users every week, most of the bugs have been shaken out of it, though the MIME code is new and has not been heavily used yet.

Domain names

Pine needs to know the full host and domain name of the site it is running on.

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

"olive.cac.washington.edu"

In this name the top level is "edu", indicating it is part of an educational institution; the second level is "washington" indicating the University of Washington; "cac" is a specific department; and "olive" is the hostname. The top level names are assigned by Internet organizations, and other names are assigned at the appropriate level. The Domain Name Service, DNS, or named on UNIX systems is the distributed data base used to look up these names. The file /etc/hosts usually sets the name of the local host. If you are not on the Internet or other network you can probably just use a single part hostname)

Pine determines the domain name it uses as follows. If no settings are made in the users .pinerc or in /usr/local/lib/pine.conf then it uses the domain name looked up from the system. This usually comes out of /etc/hosts, DNS or the local configuration data base (e.g. Yellow Pages). If this look up fails and there are no other settings Pine will exit with the error message "No host name or domain name set."

Often it happens that the domain name that Pine gets from the hostname lookup is not the fully qualified domain name because of the way the /etc/hosts file is set up. When this occurs Pine cannot fully qualify the addresses it sends out. It isn't essential that the name be fully qualified, but it is highly desirable because it can avoid some mail routing problems. In some cases sendmail will fully qualify the addresses as it's delivering the mail. Specifically, the problem is because the entry in /etc/hosts for the local system has the unqualified name first and the fully qualified name following as an alias as follows:

```
128.95.112.99    olive.cac.washington.edu    olive
```

is preferred over

```
128.95.112.99    olive    olive.cac.washington.edu
```

The domain name may be set explicitly with the "user-domain" variable in either the user .pinerc or /usr/local/lib/pine.conf. If set in /usr/local/lib/pine.conf it affects all Pine users on the system. This setting overrides the system setting.

At the University of Washington an effort has been made to encourage host-less addresses so that users may move from one host to another without changing there e-mail addresses. This usually requires some sort of data base at the domain level to direct the mail to a particular host. In place of a data base, all mail could be directed at one host. To facilitate this you can set the "use-only-domain-name" to "yes", and Pine will strip off the host part of the address. In the example above, the host-less address would be "cac.washington.edu". This stripping is achieved by removing the first part of what is assumed to be the full hostname. The removal of the host part is only applied to the full hostname looked up from the system and not to the one set by the "user-domain" variable.

Once the domain name is determined, Pine uses it for qualifying all unqualified outgoing addresses and in the From: in outgoing mail. It is also used to determine if an address is that of the current Pine user. The full host name is also used in the internal message ID.

Pine Configuration (system wide and per user)

There are two configuration files in Pine: the system wide configuration file, /usr/local/lib/pine.conf; and the per user configuration file, .pinerc in each users home directory. The syntax of the variables or options in them is the same. In a number of cases the variables are the same too. The syntax is:

<variable> = <value>

If the value is absent then the variable is unset. To set a variable to the empty value the syntax is "". Quotes may be used around any value. All values are strings and end at the new line or the closing quote. Leading and trailing space is ignore unless it is included in the quotes. For some variables the only appropriate values are "yes" and "no". If a variable is set in the system configuration file and the per user configuration file, the per user setting takes precedence. If a local variable is unset (usually the case for most variables) the system value is used. If the system value is unset then the compiled in default value is used.

Both files may contain comments which are lines beginning with a "#". 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 comments describing each variable. All the possible variables are included in this. The best thing to do is create the file initially with this command and then edit in the values you want to set. Pine will run fine without this file.

References to UNIX environment variables may be included in the pine configuration file. The format is "$variable" or "${variable}". The character "˜" will be expanded to the $HOME environment variable.

Pine will automatically create the per user .pinerc file the first time it is run. It will write entries for each variable with comments so one knows what all the possible variables are by just looking in the file. Pine reads and writes the .pinerc file occasionally during normal operation as state, like the printer configuration, is changed. Any comments or additional lines in the file will be carried along as the file is rewritten. This will guarantee backward compatibility with old versions as new variables are added to subsequent versions of Pine. It also allows the user to put additional comments in the .pinerc. Pine always writes this file at least once when running so you can tell when a user last invoked Pine by the date on this file. This is how the pine-use program (which you can compile and use) counts up the current Pine users.

Currently, most of these variables have to be set by hand with an editor. Eventually it would be nice if Pine had a user interface to set these variables so the users don’t have to venture out with a text editor. It’s intended that only experienced users make settings with the current scheme.

Descriptions of the variables:

user-id
: This is disabled for time sharing machines, but may be used on versions for PCs.

personal-name
: Your full personal name. Normally this is taken from the accounts data base (/etc/passwd) on most systems and doesn’t need to be set, but you can over ride the systems idea of who you are with this. This is only appropriate in the user .pinerc

printer
: This is the current setting for a users printer. It’s usually the same value as the standard-printer, the personal-print-command or "attached-to-ansi". The value of this in the system configuration file specifies the default value if users don’t set their own. The setting of this variable is done by the printer selection screen in Pine.

standard-printer
: Specifies the command for printer selection number 2 on the printer menu. This is only in the system configuration file.

personal-print-command
: Only in the per user configuration. This corresponds to item 3 in the printer menu. This variable retains the personal-print-command when the printer is set to something other than item 3.

last-time-prune-questioned
: Despite it's silly name, this variable records the month the user was last asked if his 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. This is only in the per user configuration.

user-domain
: Sets the domain or host name for the user, overriding the system host or domain name. See the domain name section. This can be set for the whole system and for each user.

use-only-domain-name
: Can be set to "yes" or "no". At this point anything but "yes" means "no", except when it is unset. If set to yes one part of the hostname 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 hostname is "milton.u.washington.edu" and this variable is set to "yes", then "u.washington.edu" will be used on outgoing mail. This can be set for the whole system and per user.

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 setting of this is to open an IMAP mailbox for the inbox. For example "{marge.cac.washington.edu}inbox" will open the user's standard inbox on the mail server, marge. This can be set per user and for the whole system.

elm-style-save
: Setting this to "yes" will cause Pine to generate default folder names when saving messages similar to the way elm does. The folder will be named for the username of the sender of the message. This can be set per user and for the whole system.

header-in-reply
: This governs whether or not the header of the original message is included when including the original text of a message in a reply. This can be set per user and for the whole system.

default-fcc
: The name of the folder to which all outgoing mail goes is set here. The compiled-in default is "sent-mail". It can be set to "" to turn off saving copies of outgoing mail and can be set on a per user or system wide basis.

bugs-nickname, bugs-fullname and bugs-address
: These can only be set for the whole system and specifies an entry for the address book that is always inserted if found absent. It is a way to put the address to send requests for help to in everyone's address book so users can find it easily.

smtp-server
: When this variable is set to a host name or IP address Pine will try to drop off outgoing mail to the specified host via SMTP instead of sending it by passing it off to sendmail.

editor
: Sets the name of the alternate editor for composing mail. It will be invoked with the "^_" command.

image-viewer
: This variable names the program to call for displaying parts of a MIME messages that are of type image. It is usually set to xloadimage or xv. In a future version of Pine this configuration will be replaced by the more general "mailcap".

feature-level
: This controls the user level for Pine. It may be set to "seedling" for novice users, "sapling" for intermediate users and "old-growth" for advanced users. In Pine 3.0 there is no difference between seedling and sapling modes, and old-growth just adds the "H" command to view the full header and allows the "^_" command in the composer to invoke an alternate editor.

old-style-reply
: Setting this to "yes" will cause the signature to appear at the end of the original, included text in a reply instead of at the beginning. See details in the section on replying and signatures.

signature-file
: Names the file to be included as the signature. This defaults to ".signature".

mail-directory
: Use this to change the subdirectory that Pine puts its mail files in. The default is "mail".

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

show-all-characters
: If this is set to yes, Pine will display all 8 bits of any text received, regardless of whether it matches the character set currently in use.

new-version
: If this is on, a message explaining that this is a new version of Pine will be given. In Pine 3.02 this is on by default and should be set to "no" to turn off this message.

Mail folder reading, writing, locking and check pointing

All mail files (folders) are read and written by the c-client library. Extensive effort has gone into making this as robust as possible. The c-client provides locking against several user having a mail file open at once. This is beyond the usual UNIX locking which just guarantees that two processes won't write the file at the same time and are known as "MRC" locks, named after Mark Crispin. The second user that attempts to open the folder with Pine will get it opened read-only. Because there is generally no such locking convention on UNIX platforms this additional locking is only effective against multiple uses of Pine or other mailers using the c-client such as MailManager, ms and a few others. Berkeley mail(1), elm, mh and such will be able to open the mail folder when Pine has it open. If the mail file changes on the disk underneath Pine or the c-client, they will give up on the mail file and close it. Pine also checkpoints the mail file every so often depending on how many changes have been made and how long it has been since the last change. This minimizes the potential for loss.

With Berkeley format mail files, there are three modification operations done to mail folders: appending new messages, rewriting the file to delete messages and rewriting to changes status flags.

Appending messages is a special case and doesn't threaten the integrity of the mail already in the file. If an append can't be completed for some reason, such as a full disk, the whole append is aborted and the mail file left as is. When rewriting the mail file more care has to be taken. What the c-client, and therefore Pine, does is first calculate how much the mail file would grow if all the status flags were inserted in their longest form. Then the mail file is extended by that amount, writing nulls past the end of the file to be sure there's room on the disk and to reserve the space. If the extend was successful, the mail file is rewritten over the existing data without truncating the file. This is to make sure the disk space originally occupied by the mail file is not consumed by some run away disk-block-eating process.

The extended locking works by creating lock files in /tmp of the form ".\usr\spool\mail\joe". Flock() is then used on these files; their existence 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 Berkeley locks are also created. One could say the Berkeley locks are created within the other locks. This way new mail can be delivered into an open folder.

If a mail file is modified underneath of Pine while it 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 obey MRC locks (e.g. elm, or mail) is run while Pine has the mail folder open.

Pine can read and write mail files via NFS, but IMAP is preferred. Some modifications have been made to the way it operates on the mail files to make things work better with NFS, but they are not guaranteed to work with all versions of NFS or as reliable as writing to a standard file system. See the NFS section for more details.

Terminals and Termcap

Pine has been designed to require as little as possible from the terminal. It uses termcap or terminfo depending on how it was compiled. 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 will use 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. There is an internal maximum screen size, currently 170 columns by 200 rows. If the screen is made bigger than it, the whole screen won't all be used. These maximum values are set in os-xxx.h and can be changed, but increasing them will increase memory usage.

Memory Usage

The current Pine/C-client architecture isn't the most efficient in it's memory use. It reads in the whole mail folder as it's parsing it. It also makes in-memory copies of messages as they are forwarded, saved and such, so it doesn't operate very efficiently on messages over 1Mb, but normal operation is OK. Some work has been done already to ameliorate the situation and more is possible.

Sendmail, SMTP and mail delivery

Pine submits outgoing mail either to sendmail or to an SMTP server for delivery. When using sendmail, it writes the message to a temporary file in /tmp. Then it runs a shell in the background that runs sendmail on the file and removes the file. It's done with a shell in the background so the user doesn't have to wait for sendmail to finish. 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 mail back; and the "-oi" flag to ignore dots in incoming messages. (The dots are used to indicate things such as the end of the message).

To make Pine operate as an SMTP client, the smtp-server variable can be set to the IP address or hostname of the SMTP server. When a message is sent Pine will post the message directly to the SMTP server. This is particularly useful with Pine running on UNIX workstations with IMAP because it saves having to configure sendmail on the workstation. It is also essential for the DOS version since there is no alternative to SMTP.

The Berkeley Mail file format and locking

Here are the grungy details on how the Berkeley mail file format works. This format comes to us from the ancient UNIX mail program, /bin/mail. This program was actually used to interactively read mail at one time, and is still used on some systems as the local delivery agent. The messages in the file are separated by lines of the following format:

"From user-name WWW MMM DD HH:MM:SS YYYY\n"

"User-name" is from the envelope and the date is as printed out by ctime(). This separator must be at the beginning of a line and should have a blank line before it. The mail file usually ends in "0. Because of the format of the separators, no lines in the mail message can begin with "From ", space included, so they are modified to be ">From ". You'll see this occasionally in mail messages. The date is the date the message was first received and should be carried along if the message is copied to another folder. This format is antiquated and should be replaced!

There are two kinds of locking that are commonly done. The older form creates a "xxxx.lock" file in /usr/spool/mail to lock the mail box "xxxx". This makes use of the fact the directory operations are atomic in UNIX. It also mostly works across NFS. There are involved algorithms used to determine if a lock is held and should be broken. Usually the calculation takes into account the load on the system.

The other locking scheme uses the flock() system call on the mail box. 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 4.3bsd and related machines.

These locks only control actual reading and writing to the file. When new mail is being delivered, messages are being deleted or status being updated these locks are used. For small mail files they should never be around for more than a few seconds. This is normally the only locking used for mail on UNIX mail system. The c-client/Pine extends this locking to prevent multiple mail readers from opening the same mail folder for modification twice. See the section on "Reading, Writing, Checkpointing and Locking".

Accessing mail folders via NFS

It is possible to access NFS mounted mail folders with Pine, but there are some drawbacks to doing this. One is that the per folder "MRC" locks don't work because /tmp is usually not shared, and even if it was, flock() doesn't work across NFS. This is the lock added by Pine and not the standard UNIX mail file lock against writing.

The implementation of the standard UNIX ".lock" file locking has been modified to work with NFS as follows. Standard hitching post locking is used: First a uniquely named file is created. Usually something like "inbox.host.time.pid". Then a link to it is created named "inbox.lock" where the folder being locked is "inbox". This file constitutes the lock. This is a standard UNIX locking scheme. It works because the link(2) system call is atomic. With NFS, link(2) is still guaranteed to be atomic, but packets can get lost and cause problems. The worst case happens when the link request makes it to the server and the link is actually made on the server, but the acknowledge packet gets lost. The client then retransmits the link request. When it arrives at the server, the server doesn't know it's not a new request (NFS is stateless) and returns saying the link already exists, so the whole link operation fails. The mail software will conclude the mailbox is locked and that it should wait.

This has been worked around by ignoring the return code from link(2). 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.

Another problem occurs with data caching. On many NFS implementations the client will never see new data appended to a file if the file is held open. Even a stat on the file will not reveal the new file size or modification time. This is quite different from standard UNIX file system semantics. To get around this, the c-client library reopens the file every time it checks for new mail.

One last annoyance that hasn't been solved happens when the c-client detects a change in the modification time of the mail file, but no change in the size of the file. A warning message is given to the user (which usually confuses the users). Thus far this has only been an annoyance and not actual problems with the mail file have been found.

In conclusion, it is mostly safe to access mail via NFS. The main problem will occur when two users try to access the same mail folder from different hosts. When the second user writes the file, Pine will get upset that the file has been changed underneath of it and abort operations on the folder. Pine will continue to display mail from the folder that it has in its internal cache, but it will not read nor write any further data. The only thing that will be lost out of the first session when this happens is the last few deletion marks since the last checkpoint.

MIME -- Multipart Internet Mail Extensions

Using the MIME standard, Pine is able to attach arbitrary files to mail messages. These files can be binaries, executables, spreadsheets, GIF, binhex, text files or any other sort. The receiver of the message will be able to detach the file exactly as it was sent. MIME messages may also include sounds, images and even video. Pine cannot display most of these formats, but it will identify each part or attachment and allow that part to be saved in a file.

As of these notes for Pine 3.0, MIME is a very new standard and Pine is one of the early implementations of MIME, thus many conventions aren't clear and we have probably made some big mistakes. MIME is defined in RFC-1341 and is very soon expected to be accepted by the Internet agencies as an Internet standard. There are many other folks working on MIME implementations and we expect that it's use will become widespread on the Internet. Generally, MIME is a way of encoding, in ASCII, a multipart message structure. The parts may be nested and may be of seven different types: Text, Audio, Image, Video, Message, Application and Multipart (nested). Provisions are included for encoding any binary data in ASCII in a base 64 format similar to uuencode or btoa. 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 such. There is a mechanism for splitting messages into multiple parts and reassembling them at the receiving end.

Pine is able to view and/or take apart just about any MIME message. It will display a list of all the parts, their types and sizes. All text parts will be shown unless it is not possible because of the character set. Rich text will be displayed in a very limited way (it will show bold and underlining). Pine cannot play audio messages at all (yet), nor display video. Image parts can be passed off to a program such as xloadimage to be viewed. Pine checks for the DISPLAY variable indicating it is running on an X-terminal. Pine will also display message and multipart types. 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. Messages of type message/partial cannot currently be reassembled or sent. Lastly, Pine cannot display the application type, but works well saving parts of type application/octet stream to files. (These are listed as type "file"). Any message part, regardless of type, can be saved to a file.

The main focus of the MIME in Pine has been to provide attachments thus, Pine mainly generates messages with application/octet-stream parts along with the text parts. As a bit of a bonus, Pine will recognize GIF files and tag them as images so they can be displayed on X-terminals as mentioned above. When a file is attached to Pine the file is read and evaluated as to whether it is ASCII text, 8 bit text or binary. If a file any lines longer than 500 characters in it or more than 10% of the characters 8 bit it will be considered binary. If the files has some 8 bit characters in it will be considered 8 bit text, otherwise it will be considered plain text. The first part of any multipart message generated in Pine will always be text with the other parts following.

MIME has two ways of encoding 8 bit, or non-text data known as quoted printable and base64. Quoted printable leaves most of the text alone and only changes the 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. These encodings are designed to get text and binary files through even the most improperly implemented and configured gateways intact.

Multipart messages can be forwarded and included in replies. How to do this with multimedia mail on a plain text device was not completely clear, so some improvements are possible. In both cases the parts of the original message will be included after some text. If the first part of the message was text you will be able to edit this text. If the first part of the original was not text a new blank text part will be added. At present Pine has trouble with multiply nested multipart messages and is unable to forward or reply to them. A nested multipart message with a part of type message in between will be handled correctly. Messages can be forwarded as MIME attachments rather than just text when running in old-growth mode. When this is done the original message is included with all it headers and cannot be edited. The receiver will only see the filtered header as Pine filters the headers of sub-messages.

When a user of a non-MIME mailer receives a multipart MIME messages it will look something like this:

```
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 206-543-5617
lgl@cac.washington.edu
Computing and Communications, University of Washington

--16820115-1435684063-711161753:#2306
Content-Type: APPLICATION/octet-stream; name=login
Content-Transfer-Encoding: BASE64
Content-Description: NeXT login program

AYAAAABAAAAAQAAAAQAAAL4AAAAAQAAAAEAAAJYAAAAAAAAAAAAA
AAAAAAAABfsAAADFAAAFswAAAAHAAAABwAAAAgAAAAAX190ZXh0
AAAAF9fVEVYVAAAAAAAAAAAAAAAAAAAAAAAAQpAAAAxQAAAABAAAA
AAAAAAAAAAAAABfX2Z2bWxpYl9pbml0MAAAX19URVhUAAAAAAAA
KQAAAEwAAATuAAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAF9fZnZt
XQxAABfX1RFWFQAAAAAAAAAAAAAAAAAAR1AAAAAAAAABToAAAAAgAA
AAAAAAAAAAAAAAAX19jc3RyaW5nAAAAAAAAAAF9fVEVYVAAAAAAA
BHUAAADQQAAFOgAAAAAAAAAAAAAAAAAAAAAACAAAAAAAAAAAABfX2Nv
AAAAAAAX19URVhUAAAAAAAAAAAAAAAAAAFRgAAACsAAAYLAAAAAIA
AAAAAAAAAAAAAAAAF9fZGF0YQAAAAAAAAAAAAAABfX0RBVEEAAAAA
AAVxAAAAQgAABjYAAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAX19i
AAAAAAAAF9fREFUQQAAAAAAAAAAAAAAAAAABbMAAAADAAAAAAAAAAAC
AAAAAABAAAAAAAAAAAABfX2NvbW1vbgAAAAAAAAAAAAX19EQVRBAAAA
CAlcwAlZCBMT0dJTiBGQUlMVVJFJXMgT04gJXMsICVzAHN1AGxv
Wxsb2Mgb3V0IG9mIG1lbW9yeQoAJXMgdG9vIGxvbmcNCgAvZXRj
3Vzci9hZG0vd3RtcAAAAABAKCMpUFJPR1JBTTpsb2dpbiAgUFJP
WRzLTQyICBERVZFTE9QRVI6cm9vdCAgQlVJTFQ6U3VuIE5vdiAx
zoyMSBQU1QgMTk5MAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAQCgjKSBDb3B5cmlnaHQgKGMp
DE5ODcsIDE5ODggVGhlIFJlZ2VudHMgb2YgdGhlIFVuaXZlcnNp
2FsaWZvcm5pYS4KIEFsbCByaWdodHMgcmVzZXJ2ZWQuCgBAKCMp
wk1LjQwIChCZXJrZWxleSkgNS85Lzg5AAAAABHUAAAR1f//////
wAAEdQAABHUAAAR1AAAEdQAAAEsAxwREwT/GhkSDxcWAAAR2gAA
AAR5gAAEeoAABHuAAAR8gAAEfYAABH6AAAR/gAAEgIAABIGAAAA
AAB

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

International Character Sets

Pine will pass ISO-2022 escapes sequences, though it will not read the escape character on input (yet). There is a compiled in list of about 20 of these escape sequences that are passed. Other sequences with the Escape character will not be passed. The width of these escape sequences is taken into account when displaying the text on the screen. ISO-2022 escape sequences are passed regardless of the type of text as tagged by the MIME formatting

Pine can display most character sets in ISO-8859-1 through ISO-8859-9 as well as US-ASCII, the default. Rather, Pine doesn't actually display the characters but passes them through; it is up to the actual display device to show the characters correctly. For example you will need some special terminal that can display Arabic to show ISO-8859-6 text. The character sets are:

US-ASCII	Standard 7 bit English characters
ISO-88590-1	8 bit European "latin 1" character set
ISO-88590-2	8 bit European "latin 2" character set
ISO-88590-3	8 bit European "latin 3" character set

ISO-88590-4	8 bit European "latin 4" character set
ISO-88590-5	8 bit Cyrillic
ISO-88590-6	8 bit Arabic
ISO-88590-7	8 bit Greek
ISO-88590-8	8 bit European "latin 5"" character set
ISO-88590-9	8 bit Hebrew

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

Pine makes use of the character set tags associated with text in MIME to decide if text is displayable or not. To make use of these character sets in Pine the character-set variable should be set to match the display or terminal. Most fonts on X-terminals today are ISO-8859-1. When e-mail arrives, the character set of the mail is checked and compared to the setting of the character-set variable. If it matches, the text is displayed. If it is determined that not all the character can be displayed, the ones that can be will be, and the ones that can't will be replaced by "_". For example if a French message arrives in ISO-8859-1 on a display that shows US-ASCII, the US-ASCII character will be displayed. One may set show-all-characters in the .pinerc to "yes" and all text will be displayed regardless of character set.

When mail is sent, Pine tags the outgoing text with the appropriate character set. The character set it determined by the character-set variable. If the character set is one of the ISO-8859 character sets and there are no 8 bit characters entered the tag assigned will be US-ASCII.

Section II -- Functions/Reference

Command Line Arguments

There are a few command line arguments. The order is not important:

-i	Go directly to the Mail Index, bypassing the main menu.
-sort <type>	Specify the default sort order for the main index. The type may be one of: "subject", "from", "size", "arrival", or "date". "Reverse" may be added along with any of these types.
-z	Enable ˆZ Berkeley style job control. The is normally off because it's not something that naive users may understand
-d <n>	Turn on debugging at level <n> where level <n> is between 1 and 9. The output is written to the .pine-debugX files. Levels 8 and 9 turn of cushioning of aborts like bus errors and segmentation fault
-k	Turn on function key mode.
-r	Go into restricted demo mode. Saving and exporting messages is not allowed and mail can only be addresses to oneself.
-conf	Print out a sample/fresh system wide configuration file on the standard output. This is *not* the same as per user .pinerc. They have the same syntax and some of the same variables, but they do have differences.
-f <folder>	Opens the named folder as Pine starts up. The folder name is a path relative to the users "mail" subdirectory.

-h	Provides a list, like this one, of arguments that can be given to Pine.
-nr	Goes into "anonymous news readonly" mode. This is mostly for a particular use at the University of Washington. Pine behaves appropriately for an unidentified user to read news, for example at a public information terminal. The set of commands is greatly limited and no updates to the news file status is allowed. This is usually used with both the -f and the i- flags.

Address Book

The address book is stored in the user's home directory in the file ".addressbook". The lines are of the format:

<nickname>TAB<fullname>TAB<address>

If the entry is an address list then <address> is of the format:

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

Normally entries are one per line unless it is a list and then the entry extends until the closing parenthesis. If lines are encountered in the address book that don't fit the format, those that don't have two tabs, they are ignored. An older format is also supported where the address lists don't have parenthesis. Spaces are not allowed in nick names.

Entries in the address book may refer to other entries in the address book. Lists may be infinitely nested. If addresses refer to each other in a loop this is detected and flagged. The address will be changed to "**** address loop ****".

This file is rewritten by Pine frequently in the format it thinks proper so comments or other formatting introduced with a text editor or other will not be maintained. For example, if you copy some bizarre file to .addressbook and run Pine you're likely to find the file mostly empty as Pine will only write back what it understands from the file.

There is a shell script, brk2pine.sh, that can convert Berkeley style personal aliases in .mailrc files to Pine address book format. It reads the users .mailrc by default or takes a filename as an argument and produces the Pine address book on the standard output. A good way for a user to run this is:

```
brk2pine.sh >> .addressbook
```

This will append the entries in the .mailrc to the addressbook. System administrators can convert someone else's aliases with:

```
brk2pine.sh ~joe/.mailrc >> ~joe/.addressbook
```

You can look at the shell script itself for the grungy details

The address book is kept sorted in order by the full name field. In order for this to be sensible the full names should be last name, then first name. Pine makes an attempt to encourage use of this format and will reverse the order of any names that have a single comma in them when they are in addresses on outgoing mail so it will be formatted first then last. The T, Take, command that captures addresses off incoming messages also attempts to reverse the name as it is inserted, though it doesn't always succeed. The way it works can probably be improved.

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

The Pine address book is intended to be personal rather than shared or global. It's not a good way to run a mailing list. We decided not to use or invent a scheme for a shared address book because it would only be usable by Pine users and we wanted to encourage interoperability. Pine use by one community should not preclude use of other tools. Some thought has gone into a "work group" which would allow sharing of address books. A work group would have a leader that could set up the addresses and some members that would use it. There aren't any plans to implement these "groupware" concepts though.

Sent Mail Pruning

Because the default behavior of Pine is to save a copy of each outgoing message we thought it useful to encourage the users to periodically prune their sent-mail folder. The first time a user invokes Pine each month it will offer to rename sent-mail to sent-mail-mmm-yyyy and then offer to delete all the old sent-mail-mmm-yyyy folders with an explanation of why it is offering to do so. The user will only be asked once a month and the month the user was last asked is recorded in the .pinerc file. If the user has set default_fcc in his .pinerc to "" then no renaming will be offered.

Signatures and Reply/Forward formats

The formatting of replies and forwarded messages is perhaps a matter of current style and personal taste. Pine encourages the user to put his contribution before the inclusion of the original text of the message being forwarded or replied to, though it does provide an option to have it put at the bottom. This is contrary to some conventions, but makes the message more readable. The reader doesn't have to scroll through the original text that he has probably already been seen to find the new text. If the reader wishes to see the old message(s), the reader can scroll further into the message.

When replying the user also has the option of including the original message. The original message included in a reply is always preceded by a line like:

On Tue, 14 Jan 1992, Ken Lowe wrote:

The header of the original message is optionally included if the user sets "header-in-reply=yes" in his .pinerc. Normally it is not included.

If the file .signature exists it will be included in outgoing messages. It is included before composition starts so the user has a chance to edit it out if he likes. The signature is included at the beginning of the forwarded message or reply in order to keep it with the new text that is being added. If the filename .signature conflicts with signatures used with other mail or news programs, the file name read for the signature can be changed by setting the "signature-file" variable in the .pinerc.

If "old-style-reply=yes" is included in the .pinerc then the signature will be appended to the end of the message after any included text. This is the more conventional way of including signatures.

Spelling checker

Pine and Pico use the standard UNIX spelling checker. Lines beginning with ">" (messages included in replies) are not checked. You may substitute a different spelling checker or use the usual, spell with arguments, by setting the SPELL environment variable to the command to be executed. The message text to be checked is on the standard input and the incorrect words are expected on the standard output.

Words can be added to it's dictionary. Here's a description of how it can be done on some versions of UNIX contributed by Bob Hurt.

In case anyone besides me uses the Spell-checking function in Pine (Control-T) and has words highlighted that are "correct" (spelling is relative, IMO), here's how to include a list of non-standard issue 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 and 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 root 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. I think you can do this by typing: "source .login" at the UNIX prompt instead of logging out, but if Step 3 worked above and Pine fails, try logging out. I'm fuzzy on what the command "source .login" does.

That should do it. Go into Pine and try it out. Good luck. Bob Hurt

Postponing mail

Composition of a half-done message may be postponed to a later time. Giving the ^O command in the composer does this. While a message is postponed, other messages can be composed. When the user goes into compose while a message is postponed he will be asked if he wants to continue the postponed message. Only one message may be postponed at a time. We would like Pine to be able to have more than one postponed messages, but haven't got around to it mostly because some work would have to be done to make the user interface nice. This is a good way to quickly reference other messages while composing.

In Pine 3.0 there are some problems postponing messages that have MIME components. Messages that have attachments that are local files can be postponed without any problems. The postponed message will only store a reference to the file and not the actual file, so the file should not be deleted or renamed until the message is sent. Any other attachments will be dropped. These would have resulted from forwarding or replying to a message with multiple parts. Eventually this will be fixed.

Postponing messages that use 8 bit characters sets (and therefore quoted-printable encoding) does not work correctly either. The message will not be decoded upon resumption of the composition leaving things like "=D6" in the file where the 8 bit characters where. Eventually this will be fixed.

Interrupted Mail

If the user is composing mail and is interrupted by being disconnected (SIGHUP or end of file on the standard input), Pine will save the interrupted composition and allow the user to continue it when he resumes Pine. A message will be given that an interrupted message can be continued as the user starts Pine. The user should go into compose to continue and will be asked if the interrupted message should be resumed. To get rid of an interrupted message continue composing, going all the way into the composer, and then cancel the message with ˆC.

Printers and printing

Pine can print to the standard UNIX line printers or to 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 in "attached-to-ansi" mode. This also works with most ANSI/VT100 emulators on Macs and PC's such as kermit, NCSA telnet, VersaTerm Pro and WinQVT. It has come to be a popular feature, though sometimes generating a lot of questions because the various emulators implement the print feature differently. For example NCSA telnet requires "capfile = PRN" in the config.tel file. It doesn't work at all on some others like the telnet provided with PC-NFS.

The second selection is the standard UNIX print command. The default here 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 users 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.

A small C program, ansiprt.c, is included in the contrib directory. This is can be used to get Pine to print on PostScript printers in conjunction with enscript. Once compiled and installed, ansiprt can be used in a print command like this:

```
enscript -p - | ansiprt
```

What ansiprt does, is read the standard input and write to /dev/tty. This is needed because Pine reads the standard output and error of the print command to present to the user as the result of the print job. Ansiprt also sends the ANSI escape sequences to turn the printer on and off.

Function Keys and Keymaps

The standard Pine uses alphabetic keys for most commands by default, and control keys in the composer. Despite possible appearances the current bindings are the result of much discussion and thought. (It's very unlikely we'll change them). Currently all the commands in the composer are single control characters. This keeps things very neat and simple for users. There are a few control characters left, and these are being reserved for commands to attach files to outgoing messages. Two character commands in the composer are a possibility, but we're trying to avoid them because of the added complexity for the user.

Control-S, Control-Q, Control-H and Control-\ are specifically not used for commands. This is an attempt to avoid configuration and preempt some problems. Control-H is treated the same as the delete key so, the backspace or delete key always works regardless of any configuration. This also keeps Pine out of trouble if Control-S and Control-Q are used for flow control.

Pine can also operate in a twelve function key mode. To go into this mode invoke "pine -k" or "pinef". You can link or copy the "pine" executable to "pinef" to install "pinef". The command menus at the bottom of the screen will show F1-F12 instead of the alphabetic commands. The escape sequences for the function keys correspond to those used on PC's running a locally modified version of NCSA telnet and are similar to vt100 keypad keys. They are:

	ANSI/VT100	VT52
F1:	<ESC>OP	<ESC>=a
F2:	<ESC>OQ	<ESC>=b
F3:	<ESC>OR	<ESC>=c
F4:	<ESC>OS	<ESC>=d
F5:	<ESC>Op	<ESC>=e
F6:	<ESC>Oq	<ESC>=f
F7:	<ESC>Or	<ESC>=g
F8:	<ESC>Os	<ESC>=h
F9:	<ESC>Ot	<ESC>=i
F10:	<ESC>Ou	<ESC>=j
F11:	<ESC>Ov	<ESC>=k
F12:	<ESC>Ow	<ESC>=l

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. Pine also traps escape sequences for a number of common function keys so user don't get an error message or have an unexpected command executed for each character in the function keys escape sequence. Some arrow keys on old terminals send single control characters like control-k. These arrow keys will not work with Pine (One even sends control-). The most common set is:

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 some of the function keys operate. The following are the details... Well, with Brad Greer's help, I have been successful at mapping selected keys in pine running in an xterm. I thought you might be interested in how it worked for me.

1 Start up pine from an xterm, also specifying a "resource name." This resource name will allow the user to specify resources for pine (that deviate from the defaults). -e.g., xterm -name Pine -e pine & (the resource name "Pine" has a corresponding resource section in the .Xdefaults file)

2. In my .Xdefaults file, I have the following "translations", using lower hex values:

```
Pine*VT100.Translations: #override 0
<Key>Delete:      string(0x04)     0
<Key>End:         string(0x05)     0
<Key>Escape:      string(0x03)     0
<Key>Home:        string(0x01)     0
<Key>Next:        string(0x16)     0
<Key>Prior:       string(0x19)     0
<Key>KP_Enter:    string(0x18)     0
```

These seemed to be good candidates for remapping (i.e., ease-of-use & similarity to existing mainstream software). Already, they have improved the "quality" of my life in pine.

One keymap I was not successful at was the action of deleting from the cursor to the end-of-the-line (NOT the entire line as ^K does). Often I only want to delete just part of a line, not the whole line. Any ideas? The program xedit works this way. I'd appreciate any feedback you can provide. I hope the above information proves helpful....

The following is a rough keymap for most of Pine.

KEY	MAIN	VIEW	FLDR	ADRBK	CMPSE	HDER
A	AdrBk	Attach	Add	Add	|<-	|<-
B					<-	<-
C	Cmpse	Cmpse	Cncl	Cncl	Cncl	Cncl
D		Delete	Delete	Delete	Delete	Delete
E		Export		Edit	->|	->|
F	Folders	Forward			->	->
G		Go fldr	Go fldr		Help	Help
H					Bk spc	Bk spc
I	Index	Index			TAB	TAB
J		Jump			Justify	
K					Kill ln	Kill ln
L		Print	Print	Print	Redraw	Redraw
M		Main	Main	Main	CR	CR
N	News	Next			Nxt ln	Nxt ln
O	Other	Other	Open		Postpone	
P		Prev			Prv ln	Prv ln
Q	Quit	Quit			****	****
R		Reply	Rename		Rd file	Rch hdr
S		Save	Save	Crt Lst	****	****
T		Tk Addr		Add2lst	2 Spell	2 A bk
U		Undelete			Un Del	Un Del
V	View	View			Scrll D	
W		Where	Where	Where	Where	
X		Expunge			Send	Send
Y					Scrll U	
Z		Sort			Suspend	Suspend
[					****	****
					ABORT	ABORT
]					Telnet	Telnet
‾						
?		Help	Help	Help		
–		Prev pg	Prev pg	Prev pg		
SPACE		Next pg	Next pg	Next Pg	Nxt wrd	Nxt wrd

New Mail Notification

Pine checks for new mail every 30 seconds. It only has to check the time stamp on the mail file, so doing this doesn't place a load on the system. If you really don't want to wait you can force a new mail check by attempting to move the cursor off the end of the message index three times. It'll beep and complain as you do this, but it will check for new mail on the third try.

When there is new mail, it will be shown in the index, the screen will beep and a message 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, it will show the count of new messages and the sender of the last one.

Other UNIX new mail notification methods work with different degrees of success with Pine. Most expect the mail to be taken out of the /usr/spool/mail file and then file to be truncated. Pine leaves it there so you can switch between different mailers unlike some other e-mail programs that copy all the mail to a private special-format file.

biff:

wnewmail:

csh: If the mail variable is set in the csh it will check for new mail periodically. It does this by checking the modification times on the /usr/spool/mail file. If the access time is less than the modification time and modification time and the access time are greater than the last time a check was made it prints "You have mail". The message will be "You have new mail" if the modification time is greater than the time the last command was executed.

This doesn't work perfectly with Pine because Pine modifies the mail file when it changes message status and expunges messages leading the shell to believe there is new mail.

login: The login program prints "You have mail" if there is anything in /usr/spool/mail. With Pine this is always the case, so it will always give this message. On most systems it will say "You have new mail" if the modification time is later than the access time. Though Pine rewrites the mail file when it exists to delete mail messages and update status, it resets the modification time to be earlier than the access time so this mechanism should work. When you have new mail login will say so.

Mail folder checkpointing

Periodically Pine will save the whole mail folder to disk to prevent loss of any mail or mail status in case Pine gets interrupted, disconnected or crashes. The period of time Pine waits to do the checkpoint is carefully calculated to be minimally intrusive. The timing can be changed at compile time in the os_xxx.h file and the delays are divided into levels

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 5 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 not checkpoint for 10 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 20 minutes with no check point.

Disk Quotas and Usage

All mail files are stored under the "mail" subdirectory. A command in the "other" menu shows the disk usage and free space. The total that is shown for mail folder is the total of the files found in the mail subdirectory. The shown number for the free space is either the space left in the users disk quota or the space left on the disk drive, whichever is smaller. The text associated with the message will say which. If the user is over his quota, that will be shown and a message will be displayed when the user starts up Pine. Pine knows how to access the standard UNIX quotas data bases, well at least two versions of it.

Alternate editor for composing

In Pine 3.0 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. It operates in one of two ways. If you select "old-growth" mode in your .pinerc you can type "^_" while in the composer and be prompted for the editor. If you set the "editor" variable in your .pinerc then "^_" will appear in the composer key menu and will invoke the configured editor when you type it.

We know that many folks would like to use the custom 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. There will probably be further discussion on this!...

Pico

The message composer in Pine is actually compiled in and part of Pine. Pico is a stand-alone version of that editor that can be used to edit text files or other. It has word wrap, justification and spelling just like the composer in Pine. It has the following command line options:

-f Turn on function keys

-nN Turn on new mail checking every N seconds

-v View file only, no editing

-w Disable word wrap

-z Turn on ^Z suspension

The Pico man page has more details and lists more command line options.

Mail folders and formats

Just about all mail folders are kept in the "mail" subdirectory. The one exception is "inbox" which is more of a place holder for what ever is the standard place to get new mail on the system. The default format used by Pine is the Berkeley mail format. It is used by the standard "mail" command and by elm. Basically mail messages are separated by lines that look like:

From deroest@daffy.cac.washington.edu Fri Jan 11 14:32:33 1991

If a file doesn't look like this, Pine will consider it's format to be invalid and refuse to open the file. The use of his format is also the reason you will occasionally see ">From" in the mail message text. The ">" must be inserted so there is no confusion between the text of the message and the separator. This is a silly problem that should have been solved long ago with a more reasonable mail file format. The sender and date information in the separator are not used by Pine. You can fool Pine into thinking a file is a mail folder by adding a message separator at the beginning of the file and where ever you want message boundaries. Pine believes the message header ends at the first blank line in message.

Pine also knows how to read other file formats, namely Tenex and USENET News.

The inbox folder is treated specially. It is normally kept open constantly so the arrival of new mail can be detected. The name "inbox" refers to where ever new mail is retrieved on the system. If the inbox-path variable is set, "inbox" refers to that. The "sent-mail" and "saved-message" folders are also some what special. They are automatically created if they are absent and recreated if they are deleted.

Pine provides an "export" command to write the contents of a message to a file for use outside of Pine. When a message is written this way Pine pays attention to the umask for the setting of the file permissions. When a message is saved to a folder, the folder is created so that read or write permission is granted only to the owner of the file.

Message Status and the expunge command

Each message has some status flags. Pine only uses a few of the flags supported by the Berkeley format and the c-client. A flag is set if the message is new and hasn't been read. An "N" shows in the messages index. The other flag used is "D" to mark a message as deleted. The message is not actually deleted until the mail box is expunged. The expunge operation basically deletes all the messages marked for deletion. The user may give the "X" command to do an expunge any time he wants to, or may wait till he is exiting Pine or closing the folder, when Pine offers to do the expunge. Internally, the message status is kept in the "Status:" and "X-Status:" fields in the message header.

Sorting

The mail index may now be sorted by subject, size, sender, date and arrival order. The "z" command will prompt the user for the sort order. The sort order can also be specified on the command line with the -sort flag. When a user changes folders, the sort order will go back to the original sort order, usually the arrival order. The command line sort specification changes the original sort order. Each sort order can also be reversed.

When a folder is sorted and new mail arrives in the folder it will be inserted in it's properly sorted place. This can be a little odd when the folder is sorted by something like the subject.

The sorts are all independent of case. and ignore leading or trailing white space. The subject sort ignores "Re:" at the beginning and will soon ignore "(fwd)" at the end. The sort of 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. Usually only the first sort is slow though 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.

Feature level

Pine now has a settable feature level for users of different experience. This is only partially implemented in Pine 3.0. The planned levels are "seedling", "sapling" and "old-growth". In Pine 3.0 there are only "sapling" and "old-growth" modes, and the only difference is that the "h" command to show full headers is available in old-growth mode. A command to pipe messages into other programs is next in the plan for old-growth.

Anonymous Readonly Mode for News

Pine has a mode appropriate for anonymous access to a folder or a news group. This leaves off the sender of the message in the index and allows no status updates to the folder or news group. Messages cannot be deleted or saved. Mail cannot be composed, though mail can be forwarded. This is usually used with the -i and -f options.

Section III -- Compiling, Debugging, Porting and Source Code

Debug files

If Pine is compiled with the DEBUG option on (the default in Pine 2.0), then pine will produce debugging output to a file. The file is normally .pine-debugX in the users home directory where X goes from 1 to 4. Number 1 is always the most recent session and 4 the oldest. 4 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. The default is level 2. This shows very general things and records errors. Level 9 produces rather copious amounts of output for each key stroke.

Memlog

There's a few source files and hooks in the Pine source to log each memory allocation and each free. This was used to check and debug memory leaks. The few found are fixed now. The code is retained for future checking. Two makefiles exist for compiling with the memory logging turned on for the "nxt" and "ult" platforms. This hasn't been used for a while so it might not quite work. A log of the allocations and frees is written to a file along with frequency counts and the net outstanding amount of leaked memory. Pine runs much less efficiently when this is enabled.

Signals, aborts and core dumps

Pine catches SIGHUP and does the best thing with it. If a message is under composition it will be saved as interrupted mail which can be continued by going into compose. SIGHUP occurs when the session is disconnected for reasons such as a modem hanging up, the client workstation rebooting or other. Pine also catches SIGQUIT (^\) and asks the user if he really wants to quit. This was added because we found many pine core files generated by nothing other than ^\. ^\ is retained so there is some emergency exit.

If pine is not compiled for debugging or the debug level is set low Pine will also catch the other abort signals like SIGSEGV (segmentation fault) so it can clean up the tty modes and exit. It prints a message that a bug in Pine has been encountered and dumps a core file. If the debug level 8 or higher then these signals won't be caught and the core dump will just happen.

Pine help text

The file pine.hlp contains all the help text for Pine. Normally, 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. Basically it is divided into sections, each with a name that winds up being referenced as a global variable. There is also some special formatting to keep things lined up and to allow for substitutions in the help text depending on whether the key labels are the standard alphabetic/mnemonic or they are the function keys. These files are processed by two awk scripts and turned into C files that are in turn compiled in to Pine.

This scheme can increase efficiency when Pine is compiled to have the strings as part of shared read-only text. This works on machines that have separate instruction and data space, which is most machines today 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. At the moment the Dynix port and the NeXT port do this. How to go about this for other ports will depend on the compiler. On the NeXT it is automatic. With Dynix it required some work. The file with the help text had to be compiled with the -R option. In addition, on Dynix, xstr is used to reduce all the identical strings to a single occurrence. This was done because higher performance was needed for the Dynix version because of very heavy use.

Source code organization

The source code is in four directories with a shell script, "build", that descends into each and invokes the appropriate makefile to compile Pine and related programs. The directories are:

- c-client: This is a set of library routines that form an interface to the mail store and act as a switch between different mail file formats/drivers. Currently it supports IMAP, Berkeley format (elm, mail..), Tenex and netnews. It also does all the RFC-822 parsing (a very thorough job) and is one of the pilot implementations of the upcoming Internet standard, "MIME" for multipart mail. Also, in the c-client directory is mtest, a very simple stand alone mail client that uses the c-client. It's useful for testing the c-client or imap connections independent of a more involved mail client. The c-client included here is only a minimal part of the imap source tree. If you plan on doing anything more with imap and the c-client than using or porting Pine you should get the full imap source tree from ftp.cac.washington.edu. Mark Crispin is the author of the c-client and IMAP protocol.

- pico: Pico stands for PIne's COmposer. The source here is for the Pico library and the Pico main program. When compiled with Pine it is the Pine message composer, spelling checker and such. When compiled against the Pico main it is the stand alone Pico editor. Pico is in part based on micro-emacs and is written by Mike Seibel

- imapd: The imapd daemon. This is a small amount of source to turn the c-client library into a daemon that runs on a mail server to serve imap clients. This is only needed if you are using the imap protocol.

- pine: The code here is the primary Pine user interface. It relies on the c-client and pico libraries and implements all the Pine screens except the composer. It also implements the address book, configuration files and includes the on-line help text.

- contrib: This directory contains contributed additions to Pine

Porting Pine to other Platforms

Substantial effort has gone into making Pine/Pico as portable as possible. There are still, of course, a number of machine dependencies. We've picked a small number of platforms for which we will do our best to provide a tested and running port. Currently they are Ultrix, NeXT, SunOS, Dynix and AIX/370. We are also willing to redistribute ports to other platforms as they conform to the invented scheme for handling different platforms.

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 tried 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. This is actually nine new files because there are a set of these files for the c-client, pico and pine source directories.

The make files are kept as simple and straight forward as possible because many previous attempts at automatically figuring out what to do seem to become complex and ineffective in what they set out to do: make compiling and installing the program easy. Each port is for a specific hardware/software platforms, 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, archiver 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 some cases the makefile defines a symbol named after the platform on which there can be dependencies in the source code. Pine, Pico and the C-client don't quite do everything the same (there are three authors involved). Basically to build the source in a directory for a given platform, run "make -f makefile.xxx" where "xxx" is the the three letter name of the platform.

The file os-xxx.h is used for general platform dependent #include's and #defines. A few things that affect all the source are here, such as a macro defining the type that qsort() and signal() return. All the include files that have been found to vary from one platform to another are also included here. In the case of Pine and Pico, there is only one os-xxx.h file called os-unx.h for the officially supported ports and inside it are #ifdefs on the platform specific symbol defined in the makefile. For the current set of ports, which have more in common than different, this is simpler. If the changes to os-xxx.h become large it will be better to create a new os-xxx.h file named for the new platform. We also prefer new os-xxx.c and os-xxx.c files for new ports so that we don't have to merge any source. There are a number of Pine configuration settings that are defined here such as the place it looks for certain files, defaults for the printer and folder names, the maximum screen size and so on.

The os-xxx.c file contains functions that are potentially platform dependent. Again, the idea is to gather all the dependencies in one place. Pine and Pico use the same strategy here as used with os-unx.h. Since Pine has been ported to both Berkeley and System V UNIX most of the functions for any given port have been created thus doing a port is usually a matter of finding the appropriate functions in from the existing ports. For example, there are two ways different systems do dates and time zones so you'll find two different versions of rfc822_date() in c-client/os_aix.c, and c-client/os_ult.c. The other c-client/os_xxx.c files have one or the other version of these functions.

It is strongly encouraged that no changes be made to the general source when porting and that all changes be contained to the three/nine system dependent files. 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. This scheme in Pine has been designed to avoid that, at the cost of making it a little harder to port Pine.

Test Checklist

The following is a check list for testing a new port:

___ Sending mail, check that full name is correct
___ Replying to and forwarding a message
___ Postponing message under composition
___ Make sure local user names are expanded
___ Test spelling checker
___ Catching of SIGHUP while message is composition
___ Setting of user-domain and use-only-domain in .pinerc
___ New mail notification. Should happen with Pine idle to check timeouts
___ Reading mail in index
___ Deleting and undeleting
___ Expunge to empty folder
___ Make sure that "~" expansion works
___ Save message to folder, check error conditions such as permission denied
___ Export message
___ Checkpointing (make 20 status changes, or 19 and wait 30 sec)
___ Open IMAP and RIMAP folders
___ Test opening bogus folders: invalid format, no permission
___ Open a USENET news group via IMAP
___ Pine -f option
___ Disk quotas and usage (over, under, and device space)
___ Change password
___ Lock keyboard
___ Address book operations
___ Take command
___ Send mail with empty address book
___ Make sure that SIGQUIT, ^ confirmation works (check core dump too)
___ Test panic (Give '#' command on main menu with debug level > 8)
___ Make sure SIGSTP, ^Z works (-z option)
___ Pinef
___ Sent-mail pruning
___ Printing using all three printer configurations
___ View help text & news
___ Folder list operations (rename, create, delete...)
___ Window resizing
___ Error messages for incorrect terminal types (try "foo" and "vt52")
___ Reading of /usr/local/lib/pine.conf

Section IV -- Differences between version 2.4 and 3.0

This may be a bit rough and tumble, but hopefully will provide enough information for system administrators to know what they are getting into by installing Pine 3.0 (3.02 by now).

The biggest change in Pine is the support for MIME. This has affected a lot of source code. All seems to be running well, but some of the MIME functions themselves are not highly polished. See the section on MIME for more details on what it is all about. The one case we've seen where MIME can intrude into normal non-MIME operations is when users either accidentally or intentionally include 8 bit characters in outgoing messages. This will cause Pine to apply the quoted-printable encoding to turn it into 7 bit ASCII since 8 bit characters are not allowed in Internet mail (though it often works and some make use of it). Other than the "Attchmnt:" field showing in the compose and the "A" command in the Mail Viewer, most users will not notice any differences because of the MIME support.

The Pine main menu has a bit of a new look. The "V" command has been moved off the main menu as we found something like half of the Pine users at the UW were not using the index at all. Instructors in the training classes always had to introduce folks to the index. Some users would even lose mail because they couldn't figure out where their mail was. By default, this version of Pine displays an explanation for the change in the main menu and a message that the previous version can be had with "pine.old". This can be turned off by putting "new-version=no" in /usr/local/lib/pine.conf. The "News" item was moved into the "Other" menu and the "Who to Call" text is now included as the first page of the help text on the main menu.

Pine and Pico now include a file browser for selecting files to be included "read in". The user can navigate his directory trees and such. This is invoked by typing ˆT when prompted for the file name.

New features that shouldn't have any caveat's are:

- Index can be sorted by subject, sender, size, arrival or date
- Command line option to specify sort order
- Confirms with user before sending message to the mailer-daemon
- New command line option "-i" to go directly into the index
- New command line option "-nr" for news/readonly mode
- Bcc saved in sent-mail and shown when viewed
- Fixed bug causing core dumps with very long addresses
- No limit on the number of messages in a folder
- Fix for address book screen painting problem with Procomm and with composing on VT102s at low speeds
- Passes ISO-2022 character set shift escape sequences for the display of Japanese and other languages
- Pays attention to umask when exporting messages to files
- Option in .pinerc to have signature at bottom of message
- More efficient screen painting for low speed use
- Upon opening a folder the current message is now first unread, ("N") message
- Pine.conf and .pinerc variable can reference UNIX environment variables
- Long line wrapping improved, now broken at spaces
- Checkpointing is less frequent and at more opportune times
- Carries received date along properly for Berkeley mail file format
- Disable ˆZ suspension when keyboard is locked
- Shows READONLY or CLOSED with the folder name in title bar
- When viewing mail, the header is displayed in the canonical "Full Name <address> format". Order of header is also uniform.
- Title bar format slightly different -- shows "Folder: "

There is now, or will be different modes for different user levels. Three modes are planed: "seedling", "sapling" and "old-growth". Currently, Pine defaults to "sapling" mode. There is also "old-growth" mode that adds the "h" command to view the full header and the "^_" command in the composer to select an alternate editor. These modes are selected with the user-level variable in the .pinerc.

There are a number of new configuration variable for pine.conf and

smtp-server
editor
image-viewer
feature-level
old-style-reply
signature-file
mail-directory
character-set
show-all-characters
new-version

One can now use their favorite editor to compose mail in Pine! If the editor variable is set in the .pinerc the "^_" command to start the editor will show in the composer. Alternately, when operating in old-growth mode, the "^_" command can be given, even it it's not displayed. You will be prompted for the editor of your choice.

There are now ports for Dynix/PTX and HP/UX. These are both System V based and Pine is generally more portable to System V platforms.

- Pine Technical Notes -

Version 3.0, July 1992

(Note: This is not intended to be a complete document, only helpful notes)